On the performance of emerging wireless mesh networks

Jiva Nath Bagale

Sustainable Computing Research Group
School of Computing and Technology

University of West London

A thesis submitted in fulfilment of the requirements for the degree of

Doctor of Philosophy

September, 2015

I would like to dedicate this thesis to my wife, beloved parents, brother and sweet daughter for their continuous inspiration, support and love. I am really thankful to my parents for helping me to get the best education possible and it would not have been possible to achieve this without you.

Declaration

I, Jiva N. Bagale, hereby declare that except where specific reference is made to the work of others, the contents of this dissertation titled, 'On the performance of emerging wireless mesh network', are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other University. This dissertation is the result of my own work and includes nothing which is the outcome of work done in collaboration, except where specifically indicated in the text. This dissertation contains less than **40000** words including appendices, bibliography, footnotes, tables and equations.

Jiva Nath Bagale
September, 2015

Acknowledgements

Foremost, I would like to express my sincere gratitude to my supervisor Dr. John Moore for the continuous support during my Ph.D. study and research, for his motivation, inspiration, enthusiasm and knowledge. I could not have imagined having a better support and guidance for my study as the one I received from him.

My sincere thanks also goes to my fellow Ph.D colleagues Antonio D. Kheirkhahzadeh for stimulating discussions, for inspiring to never give up and for always being there for motivation. The journey has definitely been smoother with a colleague like you.

Last but not the least, I would like to thank the University of West London and in particular the School of Computing and Technology for providing me the platform for the journey. I would like to thank Maria Pennells, senior admin officer in the graduate school, for all the administrative support throughout the degree.

Abstract

Wireless networks are increasingly used within pervasive computing. The recent development of low-cost sensors coupled with the decline in prices of embedded hardware and improvements in low-power low-rate wireless networks has made them ubiquitous. The sensors are becoming smaller and smarter enabling them to be embedded inside tiny hardware. They are already being used in various areas such as health care, industrial automation and environment monitoring. Thus, the data to be communicated can include room temperature, heart beat, user's activities or seismic events. Such networks have been deployed in wide range areas and various levels of scale. The deployment can include only a couple of sensors inside human body or hundreds of sensors monitoring the environment.

The sensors are capable of generating a huge amount of information when data is sensed regularly. The information has to be communicated to a central node in the sensor network or to the Internet. The sensor may be connected directly to the central node but it may also be connected via other sensor nodes acting as intermediate routers/forwarders. The bandwidth of a typical wireless sensor network is already small and the use of forwarders to pass the data to the central node decreases the network capacity even further. Wireless networks consist of high packet loss ratio along with the low network bandwidth. The data transfer time from the sensor nodes to the central node increases with network size. Thus it becomes challenging to regularly communicate the sensed data especially when the network grows in size. Due to this problem, it is very difficult to create a scalable sensor network which can regularly communicate sensor data.

The problem can be tackled either by improving the available network bandwidth or by reducing the amount of data communicated in the network. It is not possible to improve the network bandwidth as power limitation on the devices restricts the use of faster network standards. Also it is not acceptable to reduce the quality of the sensed data leading to loss of information

before communication. However the data can be modified without losing any information using compression techniques and the processing power of embedded devices are improving to make it possible.

In this research, the challenges and impacts of data compression on embedded devices is studied with an aim to improve the network performance and the scalability of sensor networks. In order to evaluate this, firstly messaging protocols which are suitable for embedded devices are studied and a messaging model to communicate sensor data is determined. Then data compression techniques which can be implemented on devices with limited resources and are suitable to compress typical sensor data are studied. Although compression can reduce the amount of data to be communicated over a wireless network, the time and energy costs of the process must be considered to justify the benefits. In other words, the combined compression and data transfer time must also be smaller than the uncompressed data transfer time. Also the compression and data transfer process must consume less energy than the uncompressed data transfer process. The network communication is known to be more expensive than the on-device computation in terms of energy consumption. A data sharing system is created to study the time and energy consumption trade-off of compression techniques. A mathematical model is also used to study the impact of compression on the overall network performance of various scale of sensor networks.

Contents

List of Figures

List of Tables

List of Listings

Nomenclature

Acronyms / Abbreviations

2G Second Generation

3G Third Generation

4G Fourth Generation

6LoWPAN IPv6 over Low power Wireless Personal Area Networks

AES Advanced Encryption Scheme

ARQ Automatic repeat Request

ASN.1 Abstract Syntax Notation One

CSMA-CA Carrier Sense Multiple Access - Collision Avoidance

DSM Distributed Shared Memory

EEWS Earthquake early warning systems

FEC Forward Error Correction

IEEE Institute of Electrical and Electronics Engineers

IETF The Internet Engineering Task Force

IOT Internet of Things

IPv4 Internet Protocol Version 4

IPv6 Internet Protocol Version 6

ISA International Society of Automation

JSON JavaScript Object Notation

Kbps Kilo Bit Per Second

LLNs Low-power and Lossy Networks

MAC Media Access Control

MEMS Micro-Electro-Mechanical Systems

MSS Maximum Segment Size

MTU Maximum Transmission Unit

OS Operating System

PO Packedobjects

POD PackedobjectsD

Pub/Sub Publish Subscribe

RPC Remote Procedure Call

RTT Round Trip Time

SensorML Sensor Modelling Language

SGML Standard Generalised Markup Language

SNMP Simple Network Management Protocol

URI Uniform Resource Identifier

XML Extensible Markup Language

Chapter 1

Introduction

Wireless networks have been around in different forms since the 1980s. They emerged in the form of wireless voice networks which is now commonly known as mobile networks. Wireless data networks started emerging in the 1990s in the form of second generation (2G) networks and have continued to evolve until today where third generation (3G) and fourth generation (4G) networks are about to overtake the data usage of fixed line broadband. Today, wireless networks have become even more ubiquitous and the gap between physical objects in human life and virtual world of Internet is becoming narrower than ever. The new wave of wireless sense and control is about connecting everyday objects to the virtual world in unprecedented manner. The phenomenon is referred to with different terms including *Internet of Things*, *Body Area Networks*, *Vehicular Area Networks* or *Wireless Personal Area Networks*. All of them have sensing and wireless communication in common and are different forms of Wireless Sensor Networks. The aim is to automatically monitor, control and respond to various environmental events (weather, tsunami, hurricanes or forest fires), traffic, household utility equipment and more.

The development of Micro-Electro-Mechanical Systems (MEMS) technology has facilitated the creation of small, low-cost and smart sensors. The small size of the sensors allows them to be combined with tiny embedded hardware or even used inside human body. The lower cost makes it possible to deploy large number of sensors to create a network in an economical way. The sensors have limited but powerful capabilities to collect data and perform some processing before communicating to other sensors or directly to the Internet. They are often deployed in extreme locations where it is difficult to access them regularly and thus the data is communicated over a wireless net-

work to a base station either directly or via a mesh network of similar sensors. The sensor nodes mostly rely on battery or solar power in order to operate on those difficult conditions. Thus, it is challenging to create wireless network to provide continuous data collection, processing and communication while operating on limited power resources.

Power limitation in wireless sensor networks is an ongoing research area and there have been various efforts to save power consumption in order to improve the life of the sensor nodes. The consumption has been reduced by efficiently managing the communication process (e.g. turning off radio when not in use). It is well known that network communication consumes significant energy resources in comparison to the device computation. Thus it is also beneficial to reduce the amount of data communicated over a network by processing it on the device. However the trade-off between the savings from data processing and data size reduction must be studied properly. If the data processing on the device is proved to be beneficial, it can minimise usage of the two of the most limited resources on the sensor network field, the power and the network bandwidth.

1.1 Emerging wireless networks

Wireless networks are becoming increasingly pervasive with the popularity of sensors to interact with its surroundings. The decline in price of embedded hardware and sensors has made their implementation affordable and thus more common. The development of open wireless networking standards suitable for resource-constrained systems has also been important. And as a result, wireless networks are being deployed in a wide range of areas and different levels of scale. They have been deployed in very small scale and in extremely sensitive area such as human body to track patient's heart beat, blood sugar level or blood pressure. They have also been deployed in large scale to monitor various environmental factors, industrial automation or infrastructure conditions.

The sensor data are often communicated to a central node (commonly referred as sink node) which is more powerful than the rest of the nodes and is often connected to one or more networks outside the sensor network itself. The sensor nodes are mostly connected directly to the sink node in order to transfer the sensed data. However it becomes challenging to transfer the data

to the sink node when it has to be communicated via other sensor nodes. The data transfer speed which is already low decreases further with the increase in the number of hops the data has to pass. The wireless network bandwidth degrades with packet loss in the medium and packet collisions due to a large number of wireless nodes. Thus it becomes interesting to reduce the amount of data that needs to be communicated over the network to improve the transfer time. However the quality and integrity of the data must be preserved in the process.

A typical sensor node may provide some data processing along with the task of sensor data collection and communication. Also embedded hardware have been improving a lot recently in terms of processing power, lower power consumption and the ability to run embedded OS. A minimal OS which can be run on tiny embedded devices to create wireless sensor networks allows improving the data processing features so that the sensor data can be communicated more efficiently. Sensor data are predictable in terms of data type and range and naturally consist of robust structure. This feature of the sensor data can be exploited to represent them in highly structured data formats such as JavaScript Object Notation (JSON) or Extensible Markup Language (XML). These data formats can be used for communication as well as for storage so that there is no need of additional database to store these data. It saves valuable time normally spent during data format conversion while sending and receiving these data.

The main advantage of these data formats can be achieved by defining meta-data for the sensor data. The sensor data are expected to be within pre-defined data range or of certain data type which enables them to be restricted by using a meta-data file. The processing power of the embedded device can then be used to compress the sensor data before communication. Data compression definitely reduces the amount of data communicated over network and thus eases the pressure on the limited bandwidth of the sensor network. However it must be achieved without sacrificing other valuable resources of the devices mainly time and energy. The trade-off between data compression speed, data size reduction over network and energy consumption needs to be studied to justify the advantages. If the compressed data actually arrives slower than the original data as a result of time expensive data compression process, it is not beneficial even if the data size is reduced significantly. Also if the compressed data arrives faster than the original data, it must be achieved without spending more energy than the data transfer of

the original data. It is challenging to achieve the perfect balance between data size reduction and spending the constrained resources of sensor networks in order to create a scalable data sharing system.

1.2 Motivation

This research is motivated by the potential of wireless sensor networks to be used for a wider range of scenarios to connect the physical world to the Internet and to use it to sense, control and monitor the surrounding environment, traffic or even human body. The information collected from these sensors can be very valuable to predict future environmental activities like hurricane or earthquake or to provide better health-care. The aim is to provide an efficient data sharing system which can be used to improve resource limitations of the wireless sensor networks and to enable to create low-cost large scale sensor systems.

Initial studies (Moore et al., 2012) have demonstrated the potential of low-cost sensors combined with embedded devices and low bandwidth to create economical seismic detection and recording system. These sensors have the potential to immediately provide valuable information to us which can currently be gathered only by using expensive large, often extremely expensive and complex, sensor systems. Some of these systems are so expensive that they are deployed only in some developing countries despite being known for frequent seismic activities (Allen, 2008). The alternative sensor networks have huge potential to be used where expensive systems cannot be deployed or along with them to improve the data collection. However there are challenges to be solved before such sensor networks can be easily deployed. The biggest challenge is to create a large scale network which can collect and communicate data in real-time by using the technologies available in the current market and achieve so in an economical manner. The limited processing, network and energy resources available in the sensor networks must be considered when implementing any data sharing system. In conclusion, this research is motivated by the potential of the wireless sensor networks to be used in ubiquitous computing and the challenges in implementing real-time data sharing sensor networks when using low-cost technologies.

1.3 Research approach

Wireless sensor networks have become ubiquitous as a result of the increasing number of sensors coupled with a decline in the price of hardware. They are already used in a range of areas including environment monitoring, logistic tracking and health-care management. The embedded sensors regularly produce a range of information such as temperature, location or blood glucose level. However these devices consist of limited processing capabilities, memory and bandwidth and they require running on limited power for longer period of time. It is fairly simple to implement networks with few sensor nodes (less than 5) in order to collect and communicate sensor data. The sensor nodes are often connected directly to a base station, normally a more powerful device e.g. laptop or PC, over wireless networks. The sensor nodes do not need to perform any data processing as the network is relatively less congested and scalability is not an issue. However the devices may have to connect to one or more sensor nodes before communicating to the base station as soon as the number of devices increases. A typical wireless network normally operates in a short distance range restricting the possibility of connecting all sensor nodes to a single base station. Thus, sensor nodes often form a mesh network where some sensor nodes act as intermediate relay router and one acts a bridge/gateway to connect to the base station.

The network congestion increases with the number of sensor nodes and the already limited network bandwidth is reduced even further. It becomes difficult to provide a real-time data communication system which can be deployed in a larger scale. However the sensor data can be processed on the device to reduce its size without losing the quality before communication. The data compression process reduces the network bandwidth usage but it also consumes some resources too, namely time and energy.

1.3.1 Research questions

The main research question of this thesis is focused on the data compression on embedded devices to save network resources in the sensor networks. Although it is always beneficial to reduce data size using compression techniques in terms of network bandwidth usage, the time required to achieve this must also be studied in order to validate the overall benefits. Most resources (power, network and processing) in these devices are limited and thus must

be managed carefully in order to lengthen the life of the network. In addition, the compression time may impact the overall energy consumption if it is bigger than the actual data transfer time. Similarly the energy consumption for the compression process must not exceed the energy savings achieved from the bandwidth usage reduction.

Main question

"What are the impacts of using data compression before communicating sensor data in wireless mesh networks?" This question focuses on the overall impacts of using data compression on an embedded device before communicating the sensor data over wireless mesh network. It will be evaluated by combining the results of compression and network transfer time for compressed and uncompressed sensor data with the results of energy consumption during the data compression and transfer process.

Sub questions

How much time can be saved during network transfer by applying data compression on the device over a given bandwidth and what are the possible impacts of data type or size on the time saving? The data size reduction from compression is definitely beneficial if the network bandwidth usage is considered. However the compression time may be an issue depending upon its value as it can affect real-time transmission of data and also the respective energy consumption could be higher. The compressed data must always arrive faster than the uncompressed data and it will depend highly on the available bandwidth and processing capabilities of the sending device.

Is the overall energy consumption (compression and network transfer) for compressed data smaller than that of (network transfer) the uncompressed data? The energy resources are very constrained in sensor networks and so if the data compression affects it in a negative way the advantages on network bandwidth usage and scalability will be meaningless. The compression ratio could be crucial along with the compression time for the data compression process to be energy friendly for the sensor networks.

1.3.2 Aims

The aims of this research work are to explore compression of highly structured sensor data on embedded devices and its impact on the overall network performance and to evaluate the energy consumption trade-off between communication of compressed and uncompressed data on such networks.

1.3.3 Objectives

- Review existing wireless technologies suitable for networking embedded devices.
- Explore existing messaging protocols and their suitability for embedded devices.
- Develop a prototype testbed to determine the best protocol candidate.
- Develop a light-weight data sharing system using the best messaging protocol.
- Explore existing structured data compression techniques suitable for embedded devices.
- Conduct an experiment by transferring compressed and uncompressed structured data to evaluate the compression time, compression ratio and the throughput of the network.
- Conduct an experiment to measure/estimate energy consumption of data compression and network communication.
- Use a mathematical model to analyse the expected performance of the mesh network in large scale deployments.

1.3.4 Approach

The processing capabilities of embedded devices have been improving better than the network capacity and power requirements. Most wireless sensor networks are designed in such a way that sensor nodes send data to a sink node which does the first data filtering and processing (Tsiftes and Dunkels, 2011). The nodes do not perform any data filtering or processing so all the sensor data must always travel to the sink node. However this approach is changing with better processing and memory capabilities of the nodes coupled with demand for scalable networks. It is beneficial to process the data on individual node if the applications are interested in particular events (e.g. a noise above certain level) rather than unprocessed data (Taysi et al., 2010). Also the data collected from the sensors are mostly highly structured in nature and the nature of the data from a specific sensor is predictable. For example, a typical temperature sensor always generates data that is within a certain range and the structure is always same. Thus, it is beneficial to represent them in highly structured data format instead of treating them as plain text or a string. The data format is crucial for compression purpose because highly structured data can be compressed very efficiently than plain text data when compressed using meta-data (Moore et al., 2013).

Thus embedded devices, which are capable of doing compression on the collected sensor data, are considered instead of tiny sensor nodes which can only collect and communicate such data. The aim is to utilise existing data compression libraries along with light-weight messaging middleware software to create light-weight yet scalable wireless sensor network. Thus the embedded devices should ideally run on minimal operating system (OS) such as Linux so that the data compression can be performed on the highly structured sensor data. It can be argued that the sensor data can always be communicated to a powerful sink node which can perform all the data filtering, processing and validation but that requires a complex database to store the received data and often expensive computer to perform the processing. Therefore it can contradict with the aim of creating an economical and affordable system by using smaller low-cost sensors.

1.3.5 Methodology

The research methodology is based on the development of a light-weight data sharing model suitable for embedded devices with constrained resources which can enhance the overall performance of networks of such devices. Initially messaging models and messaging protocols are evaluated for determining the most suitable model and protocol capable of running on embedded hardware and minimal OS. Experiments are carried out to study messaging protocols for their portability to embedded hardware/OS, control data overhead per application data and simplicity of implementation. The control data overhead is measured by analysing the TCP segments when communicated over network and separating the application and control data sizes. Control data are useful to provide extra information and to improve the messaging protocol design but it is interesting to minimise the overhead specially when the actual application data sizes are small and the network bandwidth is also small.

The research focuses on highly structured text data to present various sensor and configuration information. Sensor data in image and video format is not considered but the research could apply to configuration data from such sensor networks. The nature of sensor data is structured in nature as the data generated are mostly predictable and of similar type.

A light-weight data sharing system is created by combining ZeroMQ messaging protocol with Pub/Sub messaging model and highly structured data format, XML. The dataset consists of files with different text-based data types including numerical, string, time, enumeration choice and IPv4 address. The files represent typical sensor data types along with other common data types possibly used in the context. The files range from 72 to 5142 bytes in size. The XML dataset is compressed with popular text-based, schema-based compression techniques and compression speed and ratio are recorded. Further analysis is provided to demonstrate the benefits of communicating compressed data over uncompressed data. The compression techniques are also evaluated for their energy consumption to justify the energy savings of data size reduction is not negated by extra energy consumption during the compression process. The energy is measured by connecting a smart power measurement tool to the embedded device which monitors the current drawn from it.

1.4 Contributions

This research investigates the compression of data to be communicated over wireless mesh networks in embedded devices. The study of data compression in sensor networks is not new and compression of text, image, sound and video data formats have been discussed. This research focuses on the study of sensor data in the form of highly structured text data and does not include other data formats such as image, sound and video. However the concept of compression of sensor data in highly structured text format with the use of its meta-data is relatively new. Such compression methods commonly referred as schema-aware compression has been used in traditional networks for their efficient compression ratio. And they can be beneficial for reducing network bandwidth in sensor networks too. However the usage of extra resources required to achieve such compression on sensor network must be studied carefully to determine any possible side-affects. This research evaluates the trade-off of compression time and energy consumption between compressed and uncompressed structured sensor data communication over wireless mesh networks in these devices. The result of the experiment with the testbed network is validated for scalability by using a mathematical model. All the software and the wireless testbed created during this research are made public as Open-Source software. The wireless testbed and the messaging framework can be reused by researchers to improve the research area. More specifically, the contributions of this thesis are:

1. The possibility and challenges of embedded devices to communicate data on a regular basis using wireless mesh network is studied. IP-based wireless networks are mainly focused where the devices can communicate directly to other devices on the Internet without using any intermediate network gateway. It is found that regular data communication quickly becomes challenging as the network size grows, mainly due to the low bandwidth availability and high packet loss in such networks.

2. ZeroMQ and Spread messaging protocols are studied to analyse their suitability for embedded devices with limited resources and the extra control overhead required to communicate user data is compared. They have been previously studied with the focus on the supported messaging models and their suitability on enterprise networks. Both protocols are tested by running them on a range of embedded devices and the

control overhead is compared while communicating sample sensor data in a publish/subscribe model.

3. A light-weight data sharing system for communicating **highly structured data** on embedded devices is developed. Existing systems focus on communicating a sample of the data or communicating the data at a later time to adjust to the limited bandwidth availability. However the knowledge about the data can be used to perform efficient **loss-less compression**. The amount of data can thus be reduced before communicating it over the network. It is achieved by combining XML data format, Packedobjects compression and ZeroMQ messaging protocol. The data sharing system can be reused to create sensor networks to collect, process and communicate data in structured format in a range of areas such as environmental monitoring, health care or ICT for development.

4. An experiment is conducted by transferring the compressed and uncompressed structured data to evaluate the compression time, the data transfer time and the respective energy consumption for those processes. The **time and energy consumption trade-off** between communicating **compressed** and **uncompressed** data over low bandwidth network using embedded device is then analysed. The results of the experiments can be reused to decide if sensor data compression will be beneficial for saving resources such as bandwidth and energy in wireless networks.

5. A mathematical model of the low bandwidth wireless mesh network is discussed to estimate the effective network throughput of such networks and to analyse the impact of compression on the overall performance of such networks in large scale deployments when communicating compressed or uncompressed data. The model can be reused to study such impacts in future wireless networks deployments that communicate structured sensor data.

1.5 Published materials

The following work has has been disseminated during the research so far.

1. Moore, J., **Bagale, J.**, Kheirkhahzadeh, A., Komisarczuk, P. "Fingerprinting seismic activity across an internet of things", 5th IFIP International Conference on New Technologies, Mobility and Security (NTMS 2012), May 2012

2. **Bagale, J.**, Moore, J.,Kheirkhahzadeh, A., Komisarczuk, P. "Comparison of messaging protocols for emerging wireless networks", NTMS Workshop on Wireless Sensor Networks: Architectures, Deployments and Trends (WSN-ADT 2012), May 2012

3. Moore, J., **Bagale, J.**, Kheirkhahzedah, A., “Teaching networking fundamentals with sound.” The 13th IEEE International Conference on Advanced Learning Technologies, July 2013.

4. **Bagale, J.**, Moore, J., Kheirkhahzadeh, A., and Shiyanbola, A. Towards a real-time data sharing system for mobile devices. In Next Generation Mobile Apps, Services and Technologies (NGMAST), 2014 Eighth International Conference on, September 2014

5. **Bagale, J.**, Moore, J., Kheirkhahzadeh, A., and Rosunally, Y. Energy consumption trade-offs for XML compression on embedded devices. In Sustainable Internet and ICT for Sustainability 2015 (SustainIT 2015), 2015 4th IFIP/IEEE Conference on, April 2015

These publications present some of the contributions listed in section 1.4. Paper [1] presents the early studies about potential and challenges of communicating sensor data over low bandwidth network in real-time. This paper elaborates on contribution 1. Paper [2] presents the initial evaluation of messaging models and protocols which are suitable for embedded devices with constrained resources. This paper elaborates on contribution 2. Paper [3] presents the use of the messaging models discussed in paper 2 to teach wireless networking fundamentals to students. Paper [4] presents the results from data sharing system which analyses the impact of schema-informed XML compression on the overall network performance of the wireless networks. This paper elaborates on contribution 3. Paper [5] presents the results

from energy consumption experiments which determines the impact of compression speed and ratio on the overall energy consumption of the wireless networks. This paper elaborates on contribution 4 discussed above.

Presentations/Posters

1. Bagale, J. (2011) On the performance of emerging wireless mesh networks. In UWL annual MPhil/PhD conference, 11th May 2011, London (Poster)

2. Bagale, J. et al (2011) Energy efficient location-based disaster detection and reporting system Summer school of communications (University of Edinburgh) 13th -17th June 2011
   (Awarded best collaborative research proposal)

3. Bagale, J. (2012) On the performance of emerging wireless mesh networks. In UWL annual MPhil/PhD conference, 16th May 2012

4. Bagale, J. (2012) On the performance of emerging wireless mesh networks. In PhD forum on 10th International conference on Mobile systems, Applications and services, 25th June 2012

5. Bagale, J. (2013) Real time mobile trading. In Brentford dragon's den, University of West London, 30th April 2013

6. Bagale, J. (2013) Mobile distributed real-time search system. In UWL annual MPhil/PhD conference, 22nd May 2013, London (Oral)

7. Bagale, J. (2014) Towards scalable real-time data sharing system for mobile and embedded devices In UWL annual MPhil/PhD conference, 21st May 2014, London (Poster)
   (Awarded joint best poster presentation)

1.6 Organisation of the thesis

The rest of the thesis is structured as below.
Chapter 2 introduces wireless technologies used to communicate sensor data in mesh network. The focus is on the wireless networks which can be connected directly to Internet without using any intermediary gateway. It analyses the limitations/challenges of such networks for communicating highly

structured sensor data. Then it presents results from a prototype wireless network which communicates accelerometer sensor data over low bandwidth network.

Chapter 3 presents existing messaging protocols which can be used on embedded devices and compares ZeroMQ and Spread for analysing the supported messaging models and the control data overhead required for communicating application data.

Chapter 4 presents a real-time data sharing system for mobile and embedded devices which uses the messaging protocol selected in Chapter 3. XML, a highly structured data format, and its meta-data (Schema) is discussed along with publish/subscribe messaging model. They are combined with ZeroMQ messaging protocol to create a light-weight data sharing system which is suitable for low bandwidth networks.

Chapter 5 presents the experimental results from comparison of compressed and uncompressed data communication on embedded devices over low bandwidth network. The main focus of the experiment is on the compression time, compression ratio and data transfer time and the overall time required to communicate compressed and uncompressed data is compared. The importance of compression time and ratio on the overall time is then analysed. Similarly the energy consumption during the data compression and communication processes is analysed to evaluate the trade-off between compressing with various XML compression techniques and uncompressed data transfer.

Chapter 6 discusses a mathematical model to estimate TCP throughput for a 6LoWPAN network. The impact of small application data payload, TCP fragmentation and packet loss on the overall throughput of the network is analysed in the context of such networks. The result of this model is used to recalculate the data transfer time for various data in Chapter 5.

In **Chapter 7**, the research work is concluded and the contributions, limitations and future work of this thesis are presented.

Chapter 2

Related Work

This chapter provides background information about low bandwidth wireless networks with particular focus on IP-based networks. These networks present unique challenges in terms of quantity of the data that can be communicated over them and the speed that can be achieved. In addition, various messaging models which are suitable for communicating data over low bandwidth networks are also studied. Finally, the results from a prototype experiment is presented in order to highlight the challenges in communicating sensor data about seismic activities in real-time. The next section provides background knowledge about various low bandwidth wireless networks.

2.1 Low bandwidth wireless networks

Large standalone sensors have been traditionally deployed to sense/detect various events and to transfer the sensed data to other nodes for computation and processing. These sensors are normally capable of distinguishing noise from the data but are mostly expensive and thus large scale deployments are not always feasible due to the associated cost. However the rise of MEMS technology has enabled the development of low-cost, low-power sensors and multi-functional sensor nodes (Akyildiz et al., 2002). These nodes are generally capable of doing some data computation along with data collection and can also be used to communicate data over a short distance. Thus it is possible to create a network of large number of sensor nodes to collect, process and communicate the sensed data. Although these nodes are less accurate than the larger expensive sensors, more data can be collected from large number of smaller sensor nodes and can be deployed with lower cost.

A sensor network is defined as a collection of large number of smaller sensor nodes which are deployed densely within or nearby the area to be sensed. Sensor network may consists of one or more types of sensor nodes such as thermal, visual, acoustic, infrared, magnetic and seismic which can monitor a range of ambient conditions that include but are not limited to the following (Estrin et al., 1999):

- temperature,
- humidity,
- vehicular movement,
- lightning condition,
- pressure,
- noise levels,
- heartbeat,
- air pollution levels and more

Sensor networks are becoming more and more popular and are already used in many areas such as vehicular tracking (Ahmed et al., 2010), environmental monitoring (Werner-Allen et al., 2006), industrial automation (Krishnamurthy et al., 2005), infrastructure monitoring (Kim et al., 2007) and healthcare monitoring (Chen et al., 2011; Hao and Foster, 2008; Jafari et al., 2005; Lai et al., 2013; Latre et al., 2011; Singh et al., 2009). The sensors networks are often interchangeably referred to as low power and lossy networks (LLNs) due to their association with wireless networks and often require running on limited power for long periods of time. The potentially harsh environmental condition (e.g. monitoring a remote glacier) or complexity of deployment (e.g. sensors with human body) can make it difficult to recharge or replace the power source of such nodes. Thus, it's extremely important to manage the power consumption during the data sensing and communication process.

The Institute of Electrical and Electronics Engineers (IEEE) have specified the physical layer and media access control (MAC) layer standard targeted for low-power low data rate wireless person area networks. The standard is maintained by the IEEE 802.15.4 working group which was started in 2003 (IEEE, 2003). The main features of the standard are low transmitter power,

small maximum transmission unit (MTU) size at MAC layer and low cost. They can operate on several wireless bands which includes 2.4GHz, 915 MHz and 868 MHz but 2.4GHz is used mostly as it can be used worldwide (not restricted in any region/country) and also provides better data rate. The data rate can be 20-250 Kilo bit per second (Kbps) depending on the band and mode implemented. However it does not define upper network and transport layers and thus various specifications are created by defining upper layers on top of the currently defined physical and MAC layer. These specifications can mainly be divided into category of IP-based and non IP-based standards.

Major non IP-based specifications

- ZigBee,
- WirelessHART,
- ISA100.11a and
- MiWi

The ZigBee standard builds on the IEEE 802.15.4 standard by adding networking and application support functionality (ZigBee, 2006a,b). It has been widely used in areas including home automation, industrial control and wireless sensor networks. A typical ZigBee network consists of three types of devices: ZigBee coordinator, ZigBee router and ZigBee end device. However the network cannot be connected to the Internet without using intermediate gateway which can provide application and network level conversions. Also, it is a closed standard maintained only by the ZigBee alliance which makes it almost impossible to be used with open standard software such as Linux based OS. A new specification called ZigBee IP is being developed which can support IP network without the need of gateway (ZigBee, 2013). Similarly, WirelessHART uses IEEE 802.15.4 and proprietary network layer to provide non-IP based low-power sensor network and have been mostly used for industrial process control (Song et al., 2008). ISA100.11a, developed by International Society of Automation (ISA), is similar to WirelessHART but it supports multiple network protocols by using tunnelling whereas WirelessHART only supports it own HART protocol (Petersen and Carlsen, 2011). MiWi is another protocol targeting short distance low data rate networks but is not widely used as ZigBee and WirelessHART.

IP-based specifications

- 6LoWPAN

The concept of IP-based specification was started to connect *everything* to the Internet. It allows collection of sensor data from anywhere over the Internet without using any intermediate gateway and also allow the sensor nodes to be controlled remotely. A simple network bridge or a router can be used to connect the sensor network to the Internet instead of a complex application/network conversion gateway as required by the non IP-based networks. It also simplifies the device naming and addressing model as IP can be used in all the nodes. The use of IP within the network means existing network configuration, management and debugging tools can be used Mulligan (2007). For example, Simple Network Management Protocol (SNMP) (Case et al., 1988) can be used to manage an IP-based sensor networks and it is not necessary to write a new management protocol from scratch for the same purpose. The network administrators do not need to learn any proprietary protocols when IP is used throughout.

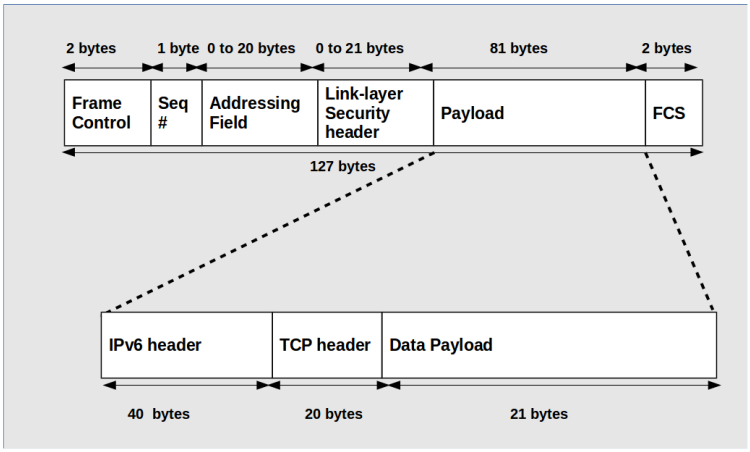

Fig. 2.1 6LoWPAN frame format

In 2005, the Internet Engineering Task Force (IETF) finalised 6LoWPAN which allows data to be communicated from IEEE 802.15.4 standard based devices to the Internet by using IPv6 (Mulligan and Bormann, 2005). Initially, there were some reservations about connecting sensor nodes and devices using IP but the 6LoWPAN working group was formed and they started to

formalise the standard (Mulligan, 2007). The main challenge of implementing IPv6 over IEEE 802.15.4 was to balance the extremely small MTU of 127 bytes and relatively large IPv6 headers of 40 bytes (Deering and Hinden, 1998). The IPv6 header consumes around 30% of the available MTU which leaves very minimum space for application data when other necessary headers are considered as shown in figure 2.1. The 6LoWPAN working group have developed a method of compressing and decompressing the IPv6 headers when they enter and leave the IEEE 802.15.4 networks (Kushalnagar et al., 2007) and has been improved further by RFC 4944 and RFC 6282 (Hui and Thubert, 2011; Montenegro et al., 2007).

6LoWPAN implementations

There are various open source implementations of 6LoWPAN standards already which offer different levels of features (Mazzer and Tourancheau, 2009; Yibo et al., 2011) and the major implementations are briefly discussed in this section. Berkeley Low power IP (BLIP) is one of the most advanced 6LoWPAN implementation and currently supports TinyOS (Levis et al., 2005). It provides most features defined in RFC 4944 namely header compression, fragmentation, addressing and neighbour discovery and supports mesh under routing (Culler, 2008a). μIPv6 is another 6LoWPAN implement which supports Contiki OS (Dunkels et al., 2004). It also provides addressing, header compression, fragmentation and neighbour discovery but does not support mesh under and route over routing (Culler, 2008b). Similarly, Nanostack, also referred as *n*stack, provides addressing, fragmentation and header compression but does not support neighbour discovery and mesh under routing but it is no longer supported Lembo et al. (2010).

BenWPAN is another 6LoWPAN implementation which is still under development and is different from others because it supports embedded Linux OS Almesberger (2012). It enables the use of 6LoWPAN network with variety of hardware such as single board computers, home routers and embedded hardware which can run minimal Linux OS. Unlike the previous implementations which can run only on specific hardware. It also enables the use of existing data processing and middleware software along with the 6LoWPAN network. In this research, it is assumed that the 6LoWPAN implementation runs on general purpose operating system such as Linux instead of TinyOS or Contiki so that existing software can be utilised.

The 6LoWPAN frame format and IPv6 header compression is explained in more detail in Chapter 6. In the next section, different messaging models which can be used for data sharing system are discussed.

2.2 Messaging models

Message passing is crucial for the functioning of distributed systems which form the basis for the Internet itself. The simplest message communication model for such systems is point-to-point where two nodes talk to each other directly. The nodes must know each other before the communication and must remain active throughout the communication cycle. The messages can be optionally stored in a virtual channel commonly known as a queue temporarily. The message is guaranteed to be delivered to one and only one receiver even if there is more than one receiver in the queue. The message remains in the queue until read by a receiver unless it expires after the message expiry time. The messaging model is synchronous in nature and request-reply, as shown in figure 2.2, is one of the most common implementation. The client sends a message to the server and waits for the reply before sending another message.

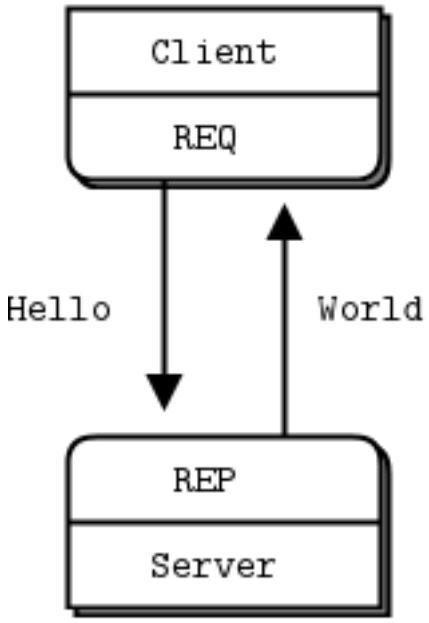

Fig. 2.2 Point to point messaging with request-reply (Hintjens, 2011)

There are different variations of the point-to-point messaging model which improve the limitations caused by the synchronous nature of the model. As shown in figure 2.3, the sender can send a message to the queue but does not expect an immediate reply from the receiver which provides some decoupling between the two nodes. Similarly, as shown in figure 2.4 the sender can send its messages on one queue and receive on another which provides decoupling between the two nodes.

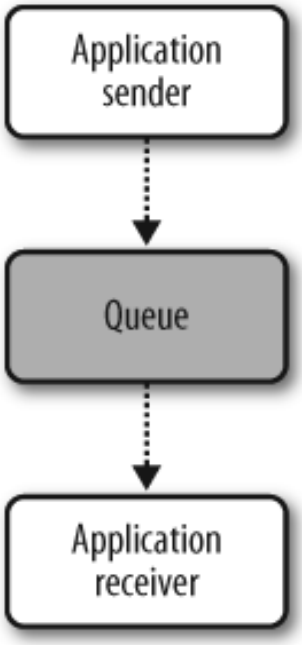

Fig. 2.3 Asynchronous point-to-point with fire and forget (Richards et al., 2009)

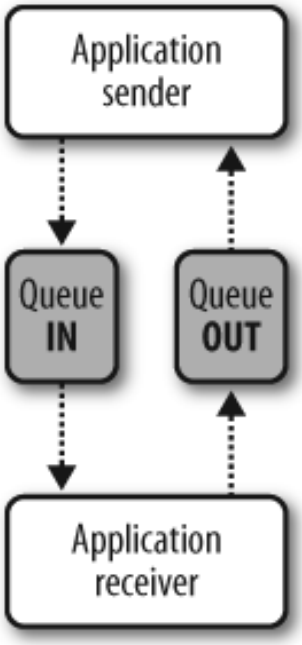

Fig. 2.4 Asynchronous point-to-point with queue in queue out (Richards et al., 2009)

However this form of messaging is static, rigid and difficult for dynamic large-scale communications mainly due to the strict coupling requirements between the sender and the receiver nodes. There are some other alternative messaging models which can decouple the communication between the two nodes and operate at different abstraction levels. **Decoupling** can be defined as the process of dissociating the sender and receiver nodes during communication. It can be performed with respect to time, space and synchronisation for messaging models. Time decoupling removes the necessity for both sender and/or receiver to be online during communication. Space decoupling means the senders and the receivers do not need to know each other before starting communication. Synchronisation decoupling means the

sender and receiver will not be blocked from performing other tasks during communication. The complete decoupling of all three factors increases the speed of the communication and decreases the complexity for large-scale communications. **Scalabilty** is referred in the coming sections/chapters of this thesis as the ability of a network to communicate more amount of data for a given bandwidth resource. The extra amount of such data might come either from existing nodes or from newer nodes in the network. The other major messaging models are Remote Procedure Call (RPC), Distributed Shared Memory (DSM) and Publish/Subscribe (Pub/Sub) (Eugster et al., 2003) and references therein.

Remote procedure call

RPC is one of most commonly used distributed interaction and was initially proposed in 1983 for remote invocation for procedural languages (Birrell and Nelson, 1984; Tay and Ananda, 1990). However it has also been used for object oriented contexts. It consists mainly of three different nodes namely *Caller, Callee and Dealer*. *Callees* register the procedures that they provide to the *Dealer* and the *Caller* invokes the registered remote procedures by using the Uniform resource identifier (URI) provided by the Dealer. The Callee then runs the requested procedure by using the supplied arguments (if any) and returns the result to the Caller (if any). The Producer (mostly the invoking node) performs a synchronous call and the consumer (mostly the replying node) processes it asynchronously. The model is not completely asynchronous as there is time coupling on consumer side and space coupling on the producer side (as instance of remote object remains active). It is possible to decouple the sender and receiver nodes in RPC to some extent as shown in figure 2.5 but it cannot be fully decoupled.

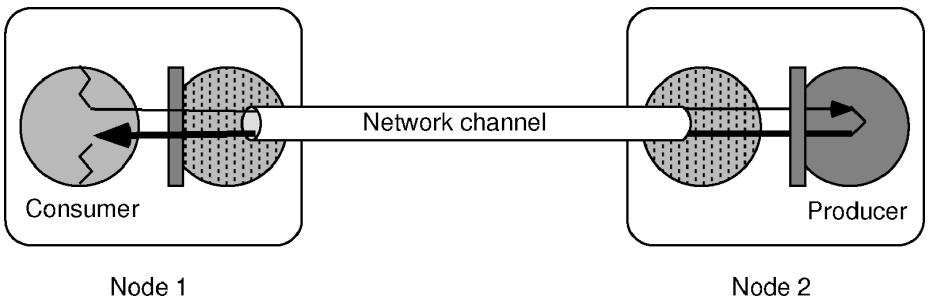

Fig. 2.5 RPC messaging model (Eugster et al., 2003)

Distributed shared memory

The concept of DSM, also referred as shared space or tuple space, started in the 1990s (Li and Hudak, 1989; Tam et al., 1990) and have been used to provide platform for accessing shared data. It is managed like an ordered queue where nodes can read, write or delete the data entries or the tuples. The model provides decoupling over time and space as the data producer and consumer remain anonymous to each other as shown in figure 2.6. However the consumers remain synchronous to the system when they pull new tuples from the space.

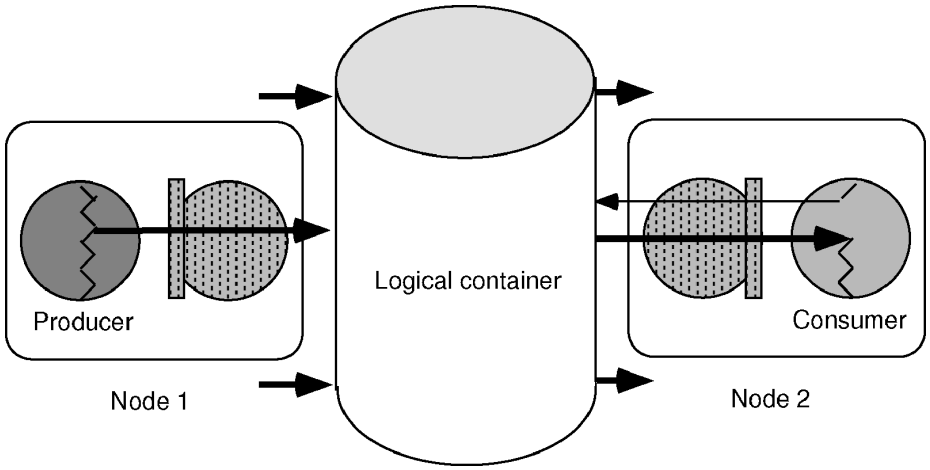

Fig. 2.6 DSM messaging model (Eugster et al., 2003)

Publish/Subscribe

The publish/subscribe message model is increasingly used in large scale or dynamic distributed environments to provide loosely coupled communication. Subscribers can register their interests in the form of an event, a pattern of events, topic or content. The interests can be registered either directly to a publisher or to an intermediate service manager as shown in figure 2.7. The publishers generate the data when there is a match for the registered interest and notify the registered subscribers via the service manager. The notification is passed from the publisher to the subscribers in completely asynchronous manner.

The breakdown of decoupling into time, space and synchronisation decoupling is shown in figure 2.8. The communicating parties do not need to know during the communication process. Both the publishers and subscribers

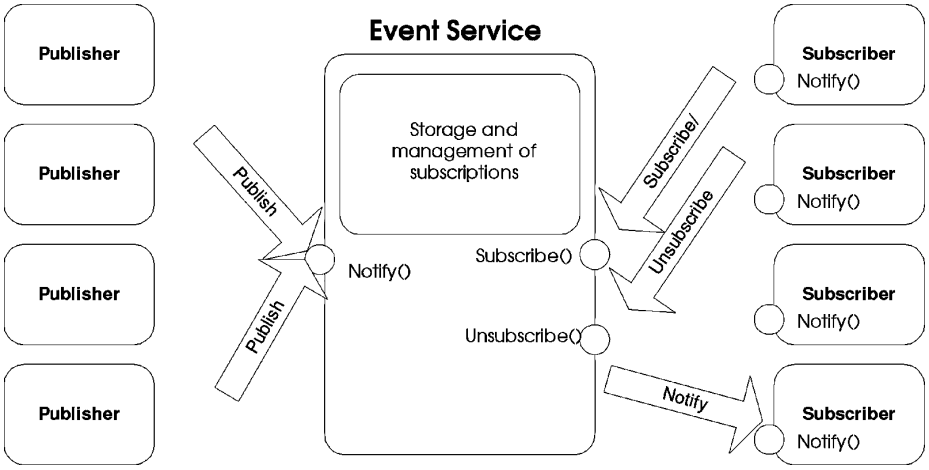

Fig. 2.7 Simple object-based publish/subscribe model (Eugster et al., 2003)

need to know only the service manager. This provides the *space decoupling* to the system as publishers do not need to know about the subscribers when sending the information and the subscribers do not need to know about the publishers when receiving the information. The communicating parties do not need to be active at the same time for the communication to be complete. The publisher can send information to the service manager and then disconnect before any subscriber connect. Then the subscribers can connect to the service manager to receive the information. This provides the *time decoupling* to the communication system. Finally, the *synchronisation decoupling* is achieved as the publishers are not blocked while sending information and subscribers can receive the information while performing some other activity.

The complete decoupling of the communication increases the overall scalability of the system as the sending and receiving parties become independent to each other. It makes the publish/subscribe messaging model suitable for distributed environments which are asynchronous by nature, such as mobile and wireless sensor networks (Huang and Garcia-Molina, 2004). The pub/sub messaging model and messaging protocols which implement the model and are also suitable for wireless LLNs are discussed in more details in **Chapter 3**. In the next section, data compression in the context of sensor networks is briefly discussed.

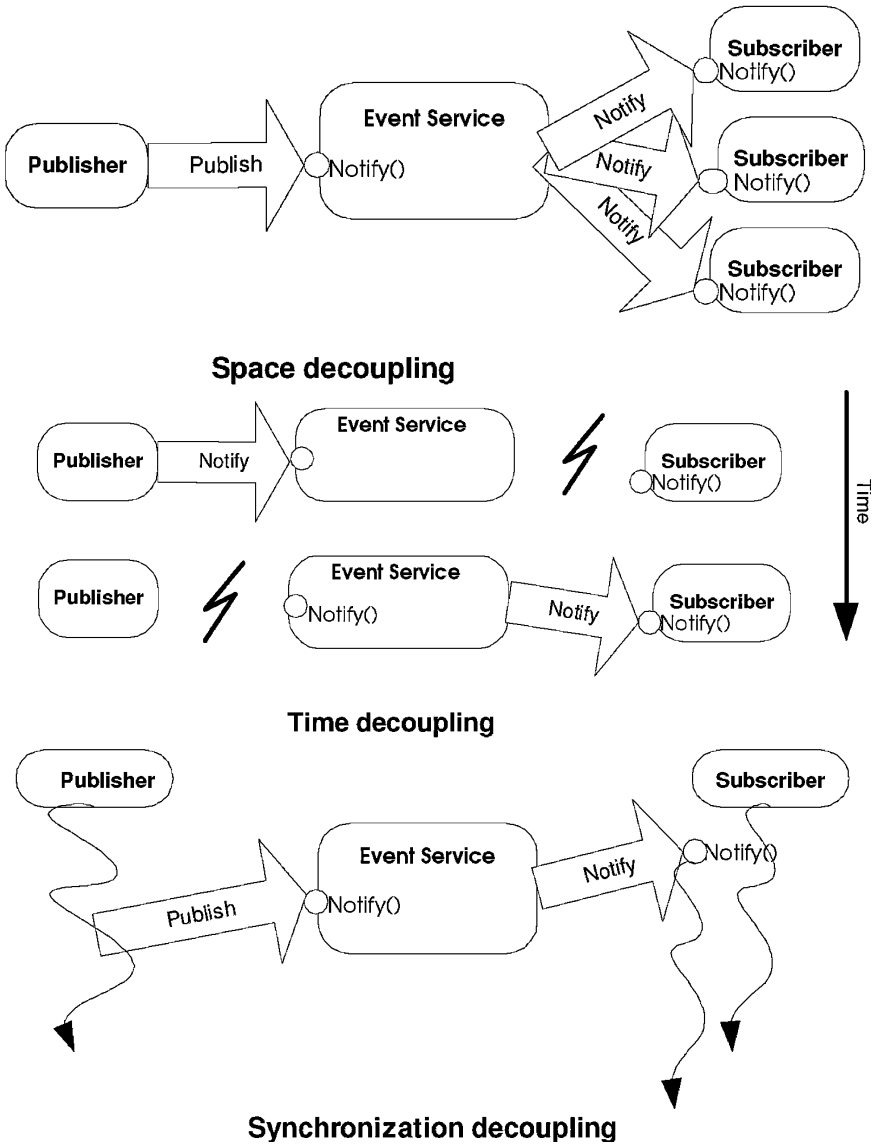

Fig. 2.8 Space, time and synchronisation decoupling with pub/sub (Eugster et al., 2003)

2.3 Data compression in sensor networks

Sensor networks can generate data in a range of formats such as text, image, sound or video. Such data can represent environmental monitoring, seismic activities, vibration waveform and acoustic signals in bridges and health-care monitoring. The sensors regularly generate data as typical data collection frequency is 100 MHz or higher. Thus the data is often processed locally before communication.

The data communication can be reduced by sampling the collected data. This approach can be used for certain data formats but may not be suitable

if the original data is required after communication. For example, lossy compression techniques have already been used to reduce data size at the expense of reduced audio and video qualities (Sinha et al., 2000). The audio and video formats can be compressed by sacrificing some quality to reduce size but the original data can still be retained. However such compression techniques can not be applied to text data format and any text-based compression technique used must be able to retain the original data. Similarly, Mammeri et al. (2012) surveyed existing image compression techniques for visual sensor networks. Authors discuss the advantages and disadvantages of discrete cosine transform, discrete wavelet transform and non transform based algorithms in the context of sensor networks. Aghdasi et al. (2008) proposed video transmission architecture for wireless sensor networks to improve energy efficiency while maintaining video quality. Ahmad et al. (2009) present improvements to existing video compression techniques, PRISM and Wyner-Ziv encoders, for sensor networks. The use of multimedia in various wireless sensor networks is surveyed by Akyildiz et al. (2007).

In this research, the focus however will be only on text-based compression of sensor data. The advantages of compression using text-based algorithm will be similar to those from image or video compression algorithms. The data size reduction provides energy savings from the fact that network communication is more energy expensive than device computation (Barr and Asanovic, 2006). However the energy and time consumption of compression algorithms must be studied carefully to determine such advantages do not come with significant trade-off. The study of text-based compression of sensor data is not new and earlier work has analysed different algorithms to study their impact on energy consumption mainly focusing on the popular implementations of Lempel-Ziv 77 (LZ77), Burrows-Wheeler Transform (BWT) and Lempel–Ziv–Welch (LZW) algorithms (Barr and Asanovic, 2006; Wang and Manner, 2009; Xu et al., 2003). These algorithms mostly rely on representing repeating sensor data efficiently to reduce size. However the knowledge of the context of sensor data can be used to reduce data size further without adversely affecting compression time or energy (Kheirkhahzadeh et al., 2013; Moore et al., 2013, 2014b). These algorithms are discussed in more detail in Chapter 5 later to determine trade-off between savings from data size reduction and extra resource cost of compression. In the next section, a wireless prototype testbed is discussed which is created using low-rate wireless network and minimal embedded Linux OS.

2.4 Fingerprinting seismic activity across Internet of things

This prototype testbed was created, firstly to check the suitability of low-cost hardware running minimal Linux OS to be used to collect sensor data, perform some data processing and communicate them over low bandwidth wireless networks. Sensor data have been previously collected from sensor nodes which have no capabilities to perform data processing or to run messaging software to be used for distributed communication. The testbed is also used to identify the challenges of running messaging software on embedded systems and of communicating large scale data over low bandwidth networks.

The aim of this prototype experiment is to use pervasive computing in order to create alternative seismic detection and recording system. The prototype is focused on using low-cost open hardware in developing countries to provide an alternative more sustainable solution to the costly infrastructure used in countries such as Japan. The work involves building a network of embedded computing devices capable of forming a broadcast group across a range of different networking technologies including emerging 802.15.4-based networks.

The ability to record seismic events over time enables utilising the data for future planning such as highlighting areas at risk from earthquakes. The data can be used to assist the infrastructure planning and design to determine suitable type and location of the buildings. These systems are already implemented in Japan and USA but rely on expensive technologies. Alternative systems which are cheaper are studied so that they can be implemented in developing countries. Most developing countries lack basic infrastructures such as broadband Internet and electricity. The alternative solutions are designed to operate over low-powered, low-bandwidth emerging networking technologies with the potential of allowing communication to take place where there is little existing infrastructure. Although the main aim is to be able to record seismic events, the feasibility of issuing immediate alerts of seismic events can be considered as well. Firstly, the problem of recording seismic events is discussed along with some related projects. Then, the architecture of such systems and the initial results are presented and the challenges of such systems are discussed.

2.4.1 Earthquake early warning systems

Earthquake early warning systems (EEWS) are not new and research has been carried out since mid 19^{th} century. In 1868, an idea was published in the San Francisco evening bulletin to deploy sensors to detect earthquake and transmit warnings over telegraphic cables and to ring earthquake bell to alert citizens (Allen, 2008). Although there was no significant progress on development of such systems for the next century, it became the basis for new research on this area. Japan has successfully deployed EEWS across their country (Nakamura, 2004). Few other countries such as USA, Mexico, Taiwan and Turkey have followed on the success of Japan on deploying these systems (Allen and Kanamori, 2003). Although California is located between two tectonic plates and is heavily studied by seismologists, due to lack of investment its EEWS is poor in hardware and unable to dispatch a warning signal in reasonable times (Allen, 2008). The same applies to other nations where earthquakes can be even more destructive due to inadequate infrastructure and the untrained population. These early warning systems use only a few powerful and expensive seismological stations to detect earthquakes. Although these systems are capable of saving lives by alerting people few valuable seconds in advance, they come at a very high price. Thus, these systems cannot be easily deployed in many developing and poor countries in the world. This led researchers to alternative approach of EEWS.

2.4.2 Embedded/Mobile earthquake early warning systems

Recent researches have focused on using small size accelerometer and deploying them in large quantity in the earthquake prone areas. The smaller accelerometer sensors are capable of detecting earthquake as soon as they occur. They are very cheap in comparison to large seismological stations and large number of accelerometers together can provide useful information about the seismic activity. This approach mostly relies on volunteers or community to provide the end computing technology as users can join the community to share their sensors data or act as listeners (Clayton et al., 2012; Cochran et al., 2009; Olivieri et al., 2008). The data is then passed to a server which analyses the data and issues warnings if necessary. The data collected can also be used to create a shake map of the earthquake affected area which can be very useful for rescue and relief operations on those areas.

But the requirement of computers and Internet access for the system has restricted implementation to the USA and European countries. However this approach has been extended (Collins and Moore, 2010; Moore et al., 2010) to mobile phones and other embedded devices with built-in accelerometers and is the focus of this prototype project. The mobile devices do not require the same infrastructure and skilled manpower as the computers and are also cheaper compared to computer and specialist seismic activity recording systems. There are increasing numbers of mobile devices which have built in accelerometer sensor and can be used to detect the seismic activity. The sensors are not restricted to expensive smart phones but are now available with low-cost feature phones as well. If EEWS can be created using mobile devices, dedicated embedded devices can be built to detect and process the seismic data. There has been very limited research on early warning systems consisting of mobile and embedded devices. It remains a challenge to communicate the earthquake shake data to other mobile devices in the network of EEWS. There is equal amount of challenge to correctly predict if any shake movement is an earthquake or not. There are chances of false warning or chance of missing real earthquake. There is no study to date to find the effectiveness of EEWS based on mobile devices. In the next section, the prototype architecture of mobile based seismic activity recording system is discussed and some initial results are presented.

2.4.3 Testbed architecture

Wireless sensor networks have been successfully deployed to create various body area networks (Latre et al., 2011). These networks usually have sensors attached to the body or any clothing of a person. The sensor data is communicated to other devices or to a server via mobile devices using wireless medium. Although the typical bandwidth available for such communication remains low the concept of these wireless networks can be utilised to form a large distributed network for EEWS. IEEE 802.15.4 working group is developing new standard targeted for wireless networks with low power consumption requirements (IEEE, 2003). Standards such as ZigBee utilise the IEEE 802.15.4 standard to create wireless network of embedded devices and have been successfully implemented on home automation systems, health care and smart energy systems. There is a new standard being developed which is known as IPv6 over low powered wireless personal area network

(6LoWPAN) which specifies various functions suitable for low powered embedded devices such as addressing, packet formats, interoperability, mesh topology and neighbour discovery (Mulligan and Bormann, 2005).

The target embedded system is primarily used to record seismic events with the possibility of issuing warnings. The target hardware should ideally have accelerometer sensors to detect seismic events. However, the software also allows devices to receive data from other devices without requiring accelerometer sensors. The software currently runs on a Linux-based PC, an Openmoko Freerunner mobile phone and a Ben NanoNote minicomputer. Additionally, the software can also run on home broadband routers such as the Buffalo G300NH which has a USB port capable of attaching an accelerometer. These different forms of hardware are connected to different groups based on their location. The groups are then connected to other nearby groups enabling sharing of data. There are various messaging protocols supporting broadcast feature required for the prototype. Protocols such as ActiveMQ, RabbitMQ, Mantaray, ZeroMQ and Spread are the most common examples. However it's not feasible to port all of them to embedded devices as required for the prototype. Spread and ZeroMQ are very simple to implement and provide C API for message broadcasting and therefore are more suitable for embedded platforms.

Each device continually collects accelerometer data at a rate based on the sampling frequency of the accelerometer used. The X-axis and Y-axis values of the accelerometer sensor are recorded during the sampling process. The changes in X and Y axes is calculated as a *covariance* which measures the changes in two or more random variables. The covariance value can be used to determine the level of shaking that is recorded by the accelerometer sensor. For example, covariance between the two axes with sample size N can be defined as equation 2.1. The covariance of every 100 samples is calculated and then compared to a predefined threshold value. If the result is greater than the threshold the device communicates an alert to the rest of the group as summarised in algorithm below.

$$Covariance(X,Y) = \sum_{i=1}^{N} \frac{(x_i - \bar{x})(y_i - \bar{y})}{(N)} \tag{2.1}$$

Where,

X_i is the value of accelerometer X-axis

Y_i is the value of accelerometer Y-axis

$\bar{X}$ and $\bar{Y}$ are average of respective axis values N is the sample size of the values

Algorithm 1 Determining accelerometer activity

```
loop
  calculate(covariance)
  if covariance ≥ threshold then
    broadcast(alertMessage) ⇒ group
  else
    donothing
  end if
end loop
```

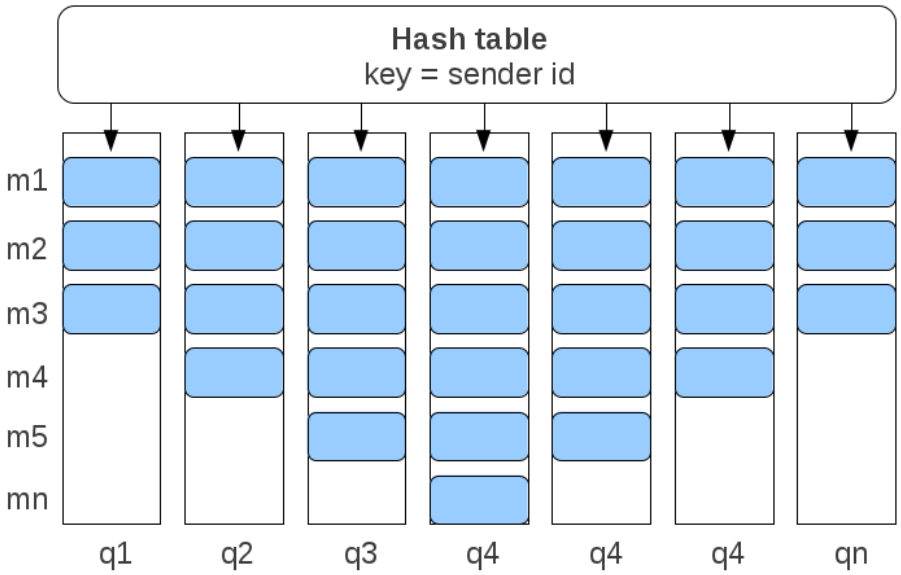

Fig. 2.9 Queuing model

Each device on the network continuously reads any messages broadcast and add them to a queue as illustrated in figure 2.9. The queues are then processed on each device every second to calculate their sizes. The queue sizes together with the sender's id are then used to calculate the frequency for queue sizes in the range $[1, 10]$. The fixed queue size range is governed by the sampling rate of the sender's accelerometer and forms the basis for the fingerprints discussed in next section.

2.4.4 Fingerprinting the network

Fig. 2.10 Ben Nanonotes and 6LoWPAN

A key part of the prototype is being able to identify significant seismic events and the process is called fingerprinting. In order to evaluate the prototype a simple message broadcast system was created using Spread messaging toolkit. A Ben Nanonote minicomputer equipped with the Ben WPAN wireless chip was used for the experiment as shown in the figure 2.10. A fake accelerometer data generator program created by Bagale (2010) is used to create sample data for the experiment. It is run on each device to provide data similar to various levels of shaking as expected to be generated from seismic activities. The data is then communicated to all other devices immediately to establish queue size and fingerprint of the event.

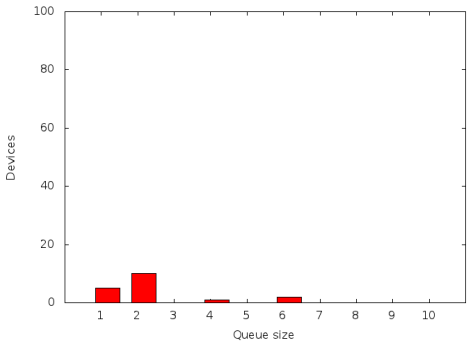

Fig. 2.11 Insignificant fingerprint

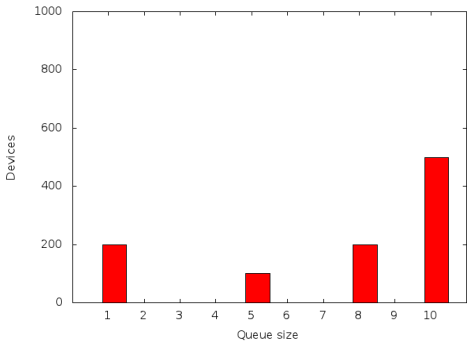

Fig. 2.12 Undetermined fingerprint

An example fingerprint is presented in figure 2.11. Here, the number of messages broadcast by each device on the network within a 1 second sample period is shown. For example, in figure 2.11 10 devices broadcast 2 mes-

sages in the sample period with little or no messages at other levels. This graph or fingerprint shows that although there was some activity on the network the amount of traffic and/or the number of devices is low, therefore, the result can ignored as being less significant. In figure 2.12, an example is shown where the fingerprint is scattered with some high and low samples size messages from significant number of machines. The pattern does not indicate any specific type of event and it is difficult to explain this pattern without knowing the circumstances for its occurrence over a period of time.

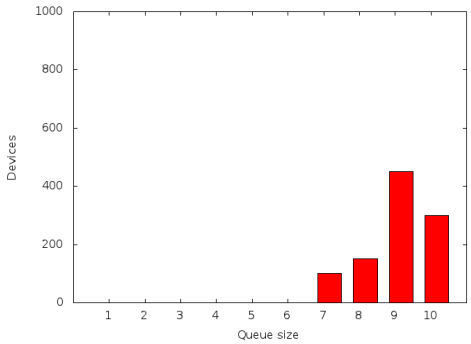

Fig. 2.13 Right-skewed fingerprint

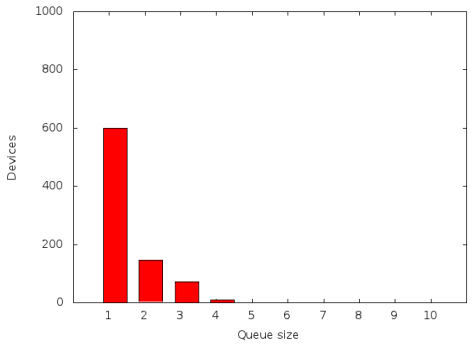

Fig. 2.14 Left-skewed fingerprint

In figure 2.13, a large number of devices are creating a right-skewed fingerprint which indicates an event occurred for a longer period of time. This kind of pattern would indicate that there has been a highly significant event based on the number of machines involved and the duration of the event. Similarly, figure 2.14 shows a large number of devices creating a left-skewed fingerprint which indicates an event occurred for a short time. This kind of pattern would indicate that there has been less significant event based on the number of machines involved and the brief duration of the event.

2.4.5 Summary of the testbed

The most important result of the testbed is to be able to use small sized low-cost accelerometer sensors to record seismic activities and to communicate them to other embedded devices over low bandwidth wireless network. The graphs discussed above prove that the sensor data can also be used to extract pattern about the actual activity although there remains a lot of improvements to be done in deriving knowledge from them and comparing to historical data. The testbed consists of around 10 embedded devices but it remains a

challenge to increase the scale of the network due to limited processing capabilities of the devices and the limited network bandwidth. The sensor data serialisation/decode process was compared among embedded devices and a personal computer (PC) to highlight the challenges present in embedded devices for processing large scale data. The major hardware specifications of those devices are listed in table 2.1.

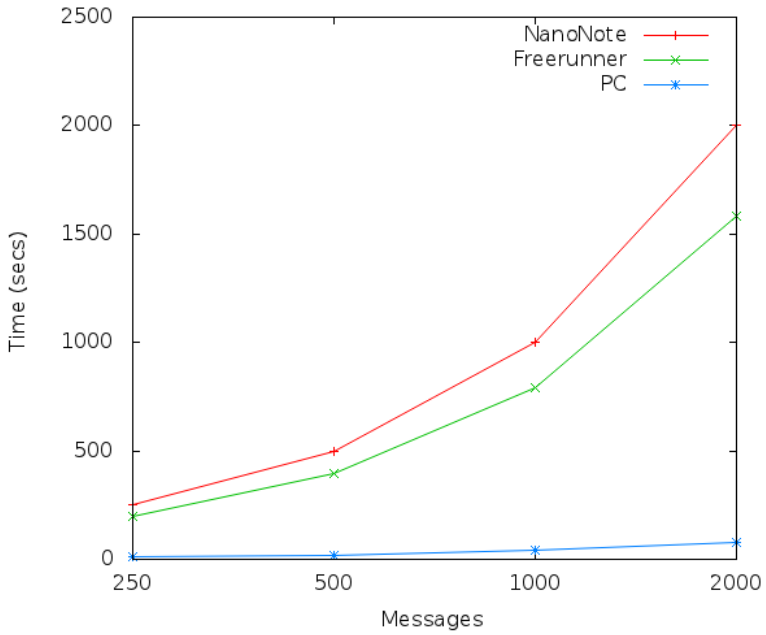

Fig. 2.15 Sensor data decode performance across embedded devices and PC

As shown in figure 2.15, the data serialisation process is extremely slow on embedded devices in comparison to a PC. It can be one of the challenges for creating a real-time large scale network as more time is required to process the data on the device. However this limitation is expected to improve with time as faster embedded devices (1GHz speed) are already becoming available on the market. It enables the improvement of the data processing performance in embedded devices.

Table 2.1 Hardware specifications for devices compared for decode performance

Device	**Processor**	**RAM**	**Architecture**	**OS**
Ben Nanonote	336 MHz	32 MB	MIPS	OpenWRT
Freerunner Phone	400 MHz	128 MB	ARM	OpenWRT
PC	1.6 GHz	1 GB	x86	Ubuntu

In the coming chapters, light-weight messaging protocols and their data processing performance on embedded devices are studied with an aim to reduce the amount of data communicated over low bandwidth networks. However this must be done without spending extra time and/or energy resources both of which are limited in sensor networks specially when communicating data in real-time.

Chapter 3

Messaging protocols suitable for embedded devices

In this chapter messaging protocols that are suitable for mobile, embedded and wireless sensor networks are evaluated. Various messaging architectures/models provided by these protocols are studied to determine their suitability. The messaging protocol needs to be suitable for continuously sending large amount of highly structured data in low bandwidth mobile and sensor networks. Although processing capabilities of the embedded devices on such networks is increasing, it is considered as well. Various messaging architectures are examined with special focus on the size of overhead per message used by these protocols in order to create a light-weight messaging system.

3.1 Background

Wireless sensor networks collect data using sensor nodes in regular interval. Data can include temperature, seismic, vibration waveform and acoustic signals in bridges (Kim et al., 2007), industrial automation (Krishnamurthy et al., 2005), environmental monitoring (Werner-Allen et al., 2006) and health-care monitoring (Chen et al., 2011). The data collected by these sensors needs to be transferred to other nodes inside the network or to devices outside the network in real-time so that it can be analysed instantly or stored for further processing. The data-rate for these sensors are typically in the range of 100Hz or higher and the amount of data increases further as the network size scales up. However it remains a challenge to transport large amount of data with constrained bandwidth and energy usage.

Sensor nodes can filter data before sending it to the network. However it might result in loss of data and it is still possible that all the sensors may try to send important data at the same time. Also significant power is consumed in a sensor node when the radio transceiver is idle (Ye et al., 2002). Thus the lifetime of the network can be improved by controlling the number of transmissions from sensors. The sensors can subscribe only to the data which is related to selected topics/interests rather than all the available topics. This approach is called data centric communication and it enables communication based on interest of receiver and not based on network address (Estrin et al., 1999). Publish Subscribe messaging systems (Pub/Sub) are common examples of such communication model. Pub/Sub model supports dynamic topologies in sensor networks and are scalable as well (Eugster et al., 2003). The information about topics can also be used to determine a path a message takes in the network and thus reducing the number of transmissions required (Rooney and Garces-Erice, 2007).

The major components of a Pub/Sub messaging system are subscribers, publishers and a broker (Hunkeler et al., 2008). Subscribers register their interest in information by subscribing to a topic. Publishers provide information to the subscribers by publishing. Broker facilitates the information delivery from publisher to a subscriber. The subscription in a typical Pub/Sub system can be done based on topic, content or type (Eugster et al., 2003). Topic based systems allow to define topics in advance which is then used to publish and subscribe information. Type based systems allow to specify the type of information to be used e.g. temperature data. Content based systems allow to describe the content of the information e.g. the light in on and temperature is below certain threshold (Hunkeler et al., 2008).

3.2 Related work

Although data recording in sensor networks is not new, traditional networks primarily focus on managing resources efficiently (Kim et al., 2007; Singh et al., 2009; Ye et al., 2002). Data recording on sensor networks can be viewed from a different perspective of data compression. Compression algorithms can be used along with existing messaging models to compress the data before sending it to the network. In Chapter 2.4 above, the testbed network discussed the performance of such messaging models. The testbed

network consisted of embedded devices such as Ben Nanonote minicomputer and a version of 6LoWPAN wireless standard (Almesberger, 2012). Ben Nanonote runs on OpenWRT Linux distribution. Tiny OS has also been used on many wireless sensor networks (Hao and Foster, 2008; Kim et al., 2007; Werner-Allen et al., 2006). However OpenWRT is used on this testbed because Linux is increasingly used on many embedded devices and also supports existing messaging and compression techniques. Although ZigBee standard is used on many sensor networks including home automation it's use has been restricted by its proprietary licensing. Thus 6LoWPAN open standard is considered more suitable for this research.

The messaging protocols are mainly of two major types, synchronous and asynchronous. Synchronous messaging is carried out in traditional client server model. Both client and server device must be available to send/receive the message and acknowledgement message must be sent before another message can be sent. Asynchronous messaging does not require the client and the server to be available at all times and immediate acknowledgement is not mandatory. It is achieved by creating a middle-ware device between traditional client and server which is commonly known as broker. The broker acts as buffer for temporarily storing messages and utilises store and forward strategy. It allows the senders to send messages before receivers are active and can disconnect once the broker receives the message as the broker can forward the message to the subscriber later. It is suitable for sensor networks as nodes can fail or disconnect from the network.

Most of the popular messaging systems use the Java Messaging Service (JMS) specification from Sun Microsystems. Examples include Apache ActiveMQ, JBoss Messaging and MantaRay and require the host machine to be able to run Java Virtual Machine (JVM). Although it is possible to run minimal JVM in embedded devices, the limited processing power and memory may be a potential drawback. ActiveMQ and JBoss messaging systems consist of a broker which facilitates the communication (Snyder et al., 2010). MantaRay is another peer to peer messaging system (Shevat, 2004). All of these systems provide robust and powerful asynchronous messaging which is suitable for large enterprise environments. However it is not feasible to implement such systems to constrained devices due to limited resources and complexity of implementation. Thus the focus is on the messaging libraries which are simple to implement and suit constrained resources available in embedded devices. Also the process of porting a messaging software to run on an embedded

device is a challenging task and must be considered too. ActiveMQ, JBoss and MantaRay have not been ported to embedded operating system such as OpenWRT yet. The messaging libraries which are being discussed below are already ported to OpenWRT. Spread Toolkit provides a unified message bus for distributed systems (Amir et al., 2004). ZeroMQ is defined as intelligent transport layer and it provides light weight messaging library for distributed systems (Hintjens, 2007).

3.3 Protocols

In this section the messaging architectures provided by ZeroMQ and Spread are discussed. The architecture which is suitable for communicating sensor data is highlighted. The selection of ideal messaging model is determined by the size of extra control data overhead required for communicating the normal data.

3.3.1 Spread

Spread is a distributed messaging system and supports the group communication model which makes it easier to build a large number of distributed applications. It provides abstraction of a group for participating nodes by facilitating message multicast between themselves. The daemons keep record of membership of the machines and pass this information to other daemons in the same group (Amir et al., 2004). It also provides message ordering and reliable message delivery within a group. It also provides failure detection of members within the group and manages group membership changes. However Spread does not provide any encryption mechanism for security.

It can be run with a daemon (server) on each local machine or with a single daemon in the network. As shown in figure 3.1 each client is running a daemon which handles the heavyweight memberships of the machines and shares the lightweight memberships of the processes to other daemons. A configuration file is used to recognise other daemons in the same network and contains the network address details of all the daemons to be grouped together. The individual daemons can serve one or more groups as shown in figure 3.1. It is possible to have up to 128 different daemons in a single LAN. However it is recommended to have less than 60 daemons in a single LAN

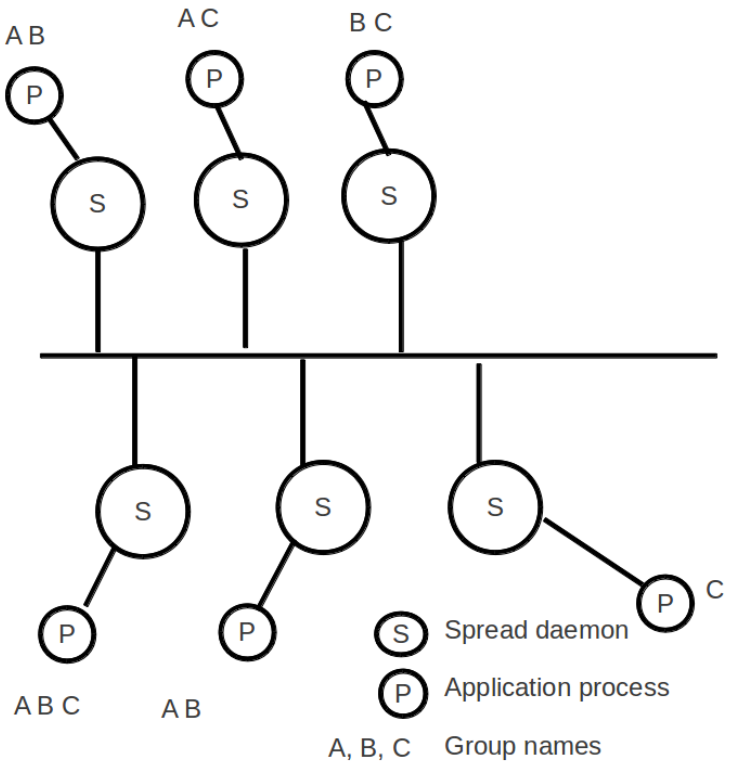

Fig. 3.1 One Spread daemon for each client

to maintain performance. Also a single configuration file can include details of up to 20 different LANs enabling the daemons to communicate with each other (Stanton, 2002).

Similarly in figure 3.2 there is a single daemon in the network which handles all the memberships and communications for all the connected clients. It is possible to connect hundreds of clients to a single daemon. However each new member in a group contributes 32 bytes extra towards the membership notification message which is sent when members join or leave the groups in that network. It causes the size of the notification message to increase and may have impact on performance of the system.

The single daemon architecture is considered to be more suitable for a typical sensor network. The configuration file requires network addresses of the machines where the daemons run in advance and the dynamic nature of sensor nodes means the changes in the configuration file can be difficult to manage. Also it requires more resources such as processing, memory and battery to run daemon in each sensor node. Thus it is beneficial to run a single daemon in the network. It is possible to create a Pub/Sub messaging

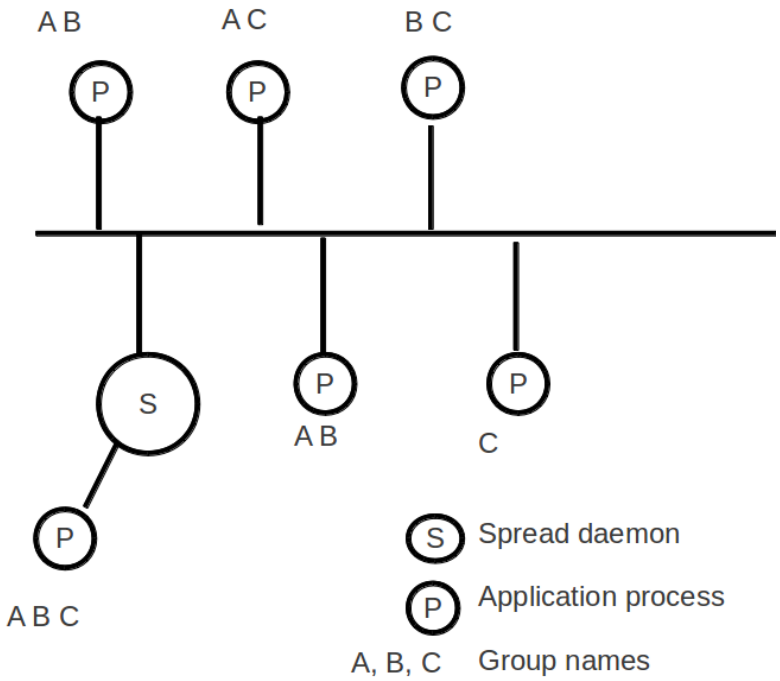

Fig. 3.2 One Spread daemon for all clients

model using single daemon architecture. The daemon acts as a broker which can receive messages from different publisher clients and then forward these messages to the subscribers based on the groups they have joined.

3.3.2 ZeroMQ

ZeroMQ is defined as the socket library which works as a concurrency framework and is a light-weight, fast and scalable messaging library (Hintjens, 2007). It has been already used to create a range of real-time applications including stock trading, network traffic monitoring and electricity monitoring applications and it has been already ported to all major platforms and consists of bindings for most programming languages. It provides messaging services using sockets which can be connected in various patterns such as publish-subscribe, request-reply and task distribution (Hintjens, 2011).

As shown in figure 3.3 request reply pattern provides one to one communication. A client sends a request message to the server and waits for response. The server then sends reply after receiving the request message.

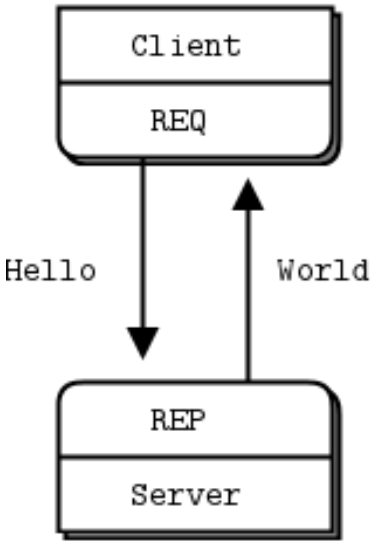

Fig. 3.3 Request reply model (Hintjens, 2011)

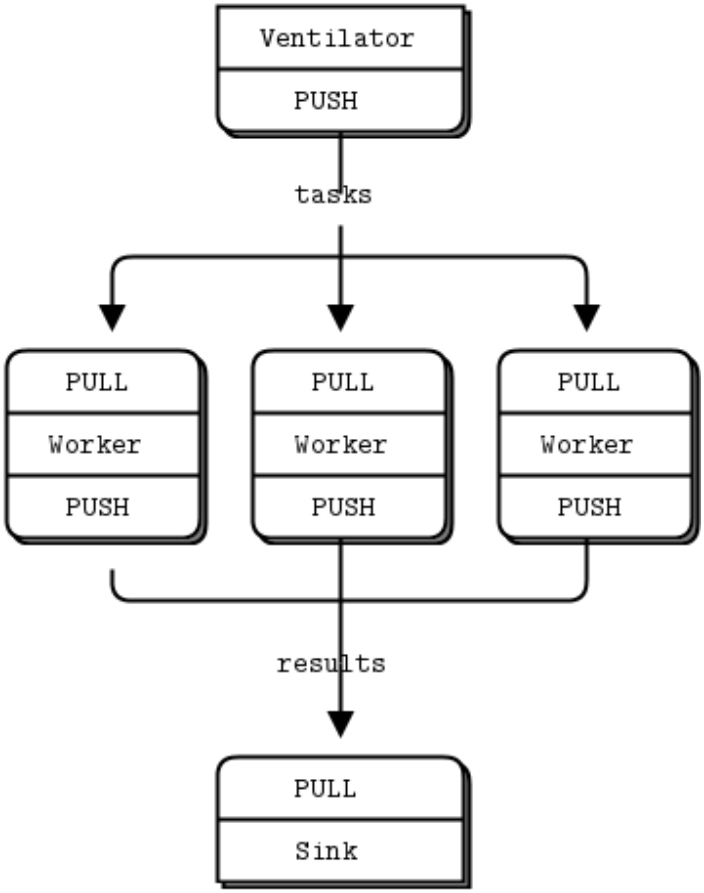

Fig. 3.4 Parallel task distribution model (Hintjens, 2011)

The client must wait for the server's reply before it can send another message. Also the server cannot send two messages in a row to the client. It is very simple communication model but it is not suitable for sending messages to multiple sensor nodes. The pipeline communication pattern shown in figure 3.4 is a parallel task distribution and collection model. The ventilator is the node which creates and pushes tasks that can be processed in parallel to the worker nodes who complete the tasks and send it to the sink node for collection of the results. This communication model is not suitable for messaging requirements of a sensor network but more suitable for distributing processing tasks to multiple machines in a hierarchical design.

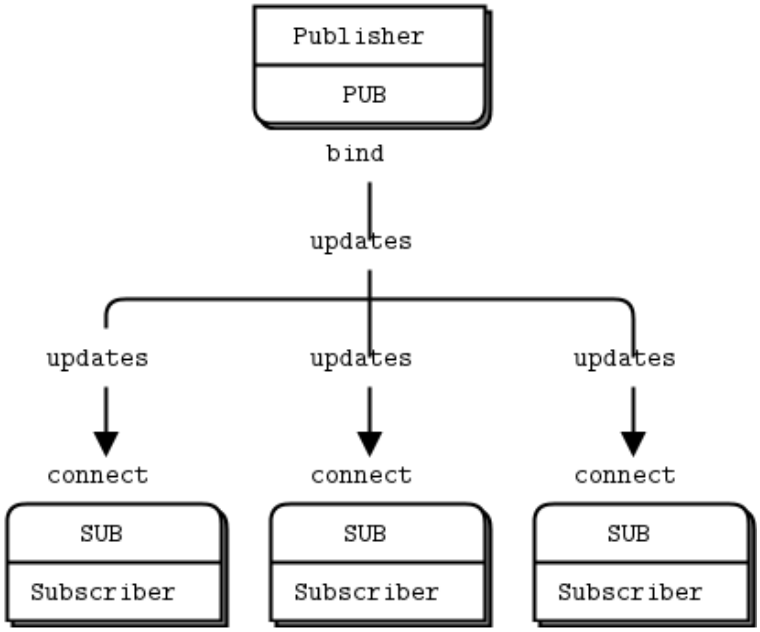

Fig. 3.5 Publish subscribe model (Hintjens, 2011)

As shown in figure 3.5 Pub/Sub model provides one to many communication. The publishers are the source of data which are sent to all the connected subscribers. The subscribers register their interest of receiving topic based data from the publisher in advance. The publisher can then continuously send messages even if there is no subscriber present. The communication is strictly one way from publisher to subscriber as the publisher cannot receive any message from subscribers. If the subscriber and the publisher have dual role for sending and receiving messages, the simple Pub/Sub model may not be suitable. However it is possible to create complex many-to-many communication model using a simpler Pub/Sub model. Each node can be assigned

to be a publisher and each node also subscribing to receive from all other publishers but it is still challenging to manage multiple many to many connections. However ZeroMQ allows creation of intermediate broker device which can act as both publisher and subscriber. It can receive message as subscriber from multiple publishers, store the message temporarily in a queue and forward it to the connected subscribers (Hintjens, 2011). Thus it is possible to create multiple publisher multiple subscriber communication using the simple Pub/Sub model.

In the next section, the single daemon architecture in Spread and multiple-publisher-multiple-subscriber architecture in ZeroMQ are compared by looking at message structure with special focus on the actual data to control data overhead.

3.4 Results and analysis

In the single daemon message architecture for Spread as discussed above the daemon maintains a message queue to temporarily store the messages received from various clients. The message queue ensures ordered delivery of the messages to all the subscriber clients who receive messages in exactly the same order as it was sent. This feature is very useful for data recording sensor networks as it ensures consistency in recorded data. A single Spread daemon can relay messages to subscribed clients from multiple groups. A client can choose to receive messages for groups A and B from the daemon while another client connected to the same daemon can choose to receive messages for groups A, B and C as shown in the figure above 3.2. Clients maintain a single connection to receive messages from all the groups they have subscribed with the daemon. The daemon must tag all the data with sending user's identity and the group information. The same connection is also used to receive membership notification messages from the daemon which can cause delay to the actual data transmission. And the notification messages affect the latency as it increases linearly with the number of members in a group increases.

Similarly a simple Pub/Sub messaging model can be expanded to create a multiple-publisher-multiple-subscriber model to fulfil the requirements of sensor networks. The intermediate broker can simplify many to many communication connections between multiple sensor nodes and allow each node to

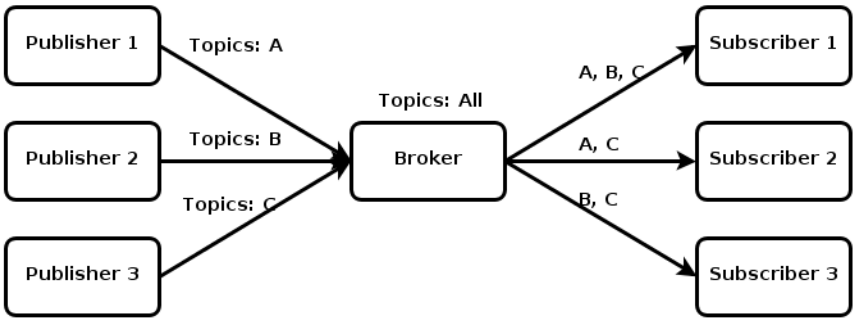

Fig. 3.6 Broker based publish subscribe model

broadcast messages to each other. As shown in figure 3.6 a broker receives messages from different groups/topics as if it is subscribed to all 3 topics A, B and C. The broker then sends those messages selectively to the different subscribers based on the interests they showed. Both publishers and subscribers contact the broker before starting communication making it easier for creating complex many-to-many communication model in a simple manner. The broker does not permanently store any message while forwarding it from publishers to subscribers and it does not pass membership notifications like Spread daemon.

The size of message sent across the network by Spread and ZeroMQ is measured using the *tcpdump* network packet analyser (Jacobson et al., 1989). The main aim of this test was to find the size of the control data added to the original message by Spread and ZeroMQ before sending it across the network. Both tests were carried out with same application program implemented in Spread and ZeroMQ messaging. An application acting as a publisher sends single byte of message to the Spread daemon and the ZeroMQ broker respectively. Another application acting as a subscriber receives the message from the daemon and the broker respectively. The Spread message consists of the group and the user id information along with the actual data. ZeroMQ does not consist of the group and the user id information by default but could be added as part of the actual data. The group and user id data can be useful to indicate sender of data and the related group or topic. The group and user id information can be considered to be common in both ZeroMQ and Spread and hence ignored while comparing overhead data.

As shown in figure 3.7 Spread adds 64 Bytes of control message when the data is sent from a publisher to the daemon. It also adds 28 Bytes of control message when it forwards the message to the subscribers. The figure

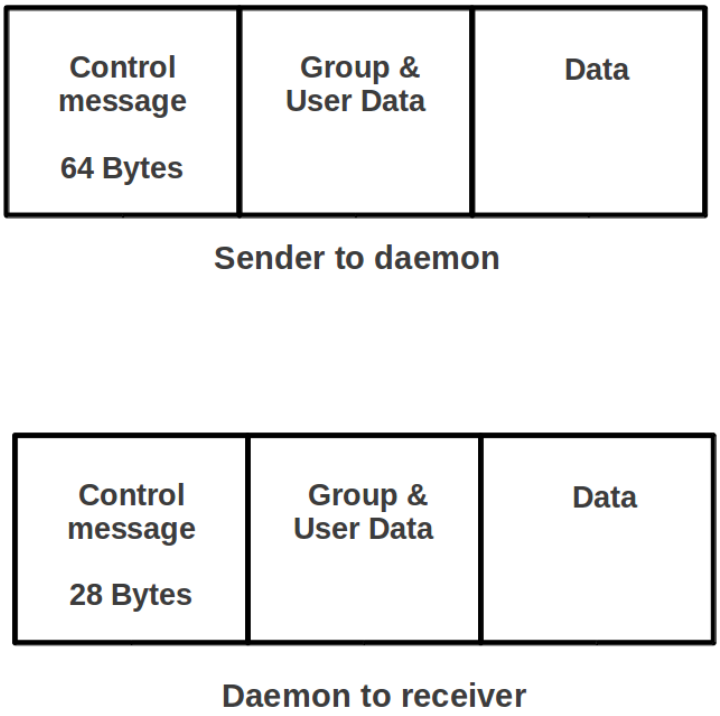

Fig. 3.7 Spread message structure

3.8 shows that ZeroMQ on the other hand adds only 2 extra Bytes of control message when the same data is sent from a publisher to the broker and another 2 bytes when data is forwarded from the broker to the subscribers. Thus Spread uses 92 Bytes and ZeroMQ uses 4 bytes on the overall for the control data overhead even when a publisher sends just a single byte of data to the subscriber.

The Spread messaging system communicates larger control data overhead and also sends extra membership notification messages along with the actual user data. ZeroMQ provides simple and light-weight Pub/Sub messaging model despite ignoring the group and user id data sent with the actual message. ZeroMQ also allows publishers to drop messages when there are not any active subscriber present preventing the data to enter the network unnecessarily. It can make significant difference on sensor networks consisting of constrained network bandwidth. Thus, ZeroMQ is considered to be more suitable messaging protocol for creating a light-weight messaging system on the embedded and sensor networks.

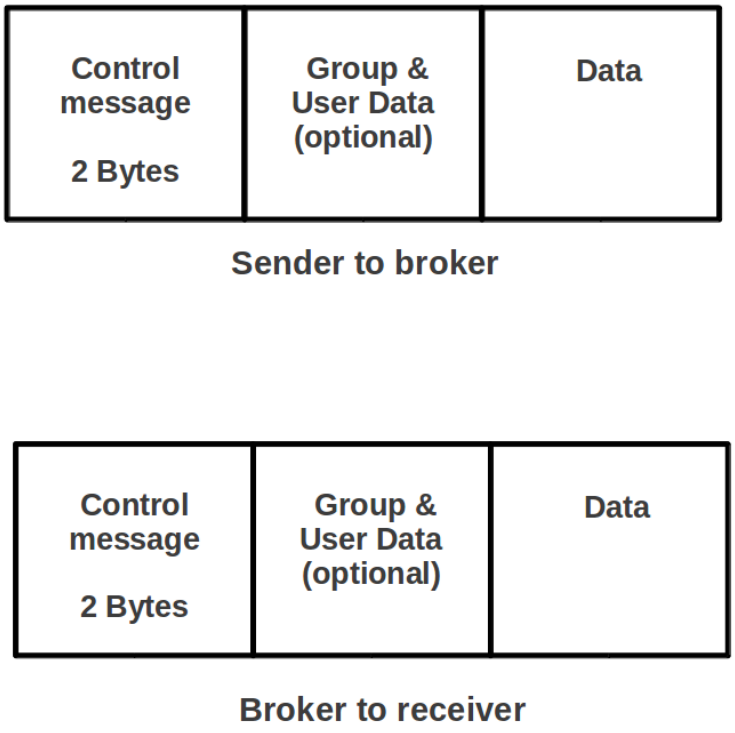

Fig. 3.8 ZeroMQ message structure

3.5 Summary

In this chapter messaging architectures provided by various messaging protocols were discussed. The comparison of these messaging protocols was performed with focus in the context of a low bandwidth sensor network. Sensor networks typically have limited processing power and bandwidth. However the sensors can generate large amount of data which needs to be communicated instantly across the network. The messaging requirement is that the devices in the group must be able to send and receive messages from each other for further data analysis, processing or storage. Messaging models provided by Spread and ZeroMQ are discussed and analysed. Spread's single daemon messaging architecture and ZeroMQ's multiple-publisher-multiple-subscriber messaging are deemed suitable for data sharing on sensor networks. Although Spread also provides group membership function, it sends more control data along with the actual data. ZeroMQ provides a simple Pub/Sub based messaging system which fulfils the messaging requirements discussed above and can also be implemented easily on embedded devices.

Chapter 4

Real-time data sharing system for mobile and embedded devices

Mobile devices are increasingly used for information sharing. The sensors embedded inside these devices are generating a range of information about their location, surrounding environment and user activities. This information can be shared with others in real-time so that it can be used or analysed instantaneously. The popularity of participatory sensing involving humans and mobile devices (phones, PDAs, tablets etc) has also fuelled the growth of large scale data management. Although the typical network bandwidth available in mobile devices has been improving it remains limited with the rise in communication activity. Therefore, data could be optimised on the device to make it more suitable for the available network bandwidth. A scalable real-time data sharing system can be built by using existing message formats, messaging architectures and compression techniques.

Bandwidth limitation, scalability issue and the impact of message compression in such networks is discussed. Compression can reduce the size of the data to be communicated by using processing power of the device. However the compression process should not take significant amount of time to be considered useful by users. In order to justify the benefits, the overall time for compressing and communicating such data must be smaller than the time for communicating the uncompressed data.

4.1 Background

Mobile devices are becoming popular for data computation along with their traditional use of voice communication. The decline in price of hardware and advances in sensor technologies such as GPS, accelerometers, gyroscopes and compasses means location-based and other data computation services have risen significantly. Thus the amount of data generated in real-time has also increased. The local storage capacity of these devices has been increasing and can be used to store the information generated by the sensors along with other user data. However it is challenging to communicate real-time data among these devices regularly as scalability becomes an issue. Various applications such as mobile TV, video conferencing, online games and other location based services already consume a significant share of the available bandwidth. Thus there is competition among applications to use limited bandwidth. On the other hand the processing power of these devices has increased drastically in comparisons to the bandwidth which can be used to optimise the data to improve the overall performance of the network.

The aim is to utilise processing and storage capabilities of the devices to create a scalable data sharing system for collecting, storing and communicating sensor data along with user data. It is on a light-weight networking infrastructure that can be run efficiently on these devices with their constrained resources and allows devices to continually search other devices for data. Each device maintains a local database in highly structured format which is accessible to others in the same group (Swaroop and Shanker, 2010). The devices are grouped together based on their interest, topic or some feature instead of their network address. A predefined meta-data (schema) is used as a network protocol for the communication and also to optimise the data before being sent over the network. The compression is performed up to bit level by using knowledge about the data and thus helps to reduce the network bandwidth usage and improve the overall performance and scalability. The system aims to run continuously in the background without interfering much with the device's normal network operations.

4.2 Related work

Most mobile distributed sharing applications focus on indexing data (Lindemann and Waldhorst, 2002; Mischke and Stiller, 2004) or algorithms to improve search time (Ley et al., 2007; Shen et al., 2012). However, most studies have not focused on optimising data to suit the limited bandwidth and to enhance system scalability. But the objective of this system is different to the existing distributed search systems as it focuses on combining the strengths of structured data, publish subscribe messaging and structured data compression in order to improve the scalability.

4.2.1 Structured data

Structured data has been widely used in web, data storage and data communication (Halevy, 2010). It is easily identifiable by computers and also readable to humans because the information is organised in hierarchy in comparison with plain data. Large amounts of data can be optimised efficiently for storage or communication. A distributed storage system for managing structured data has been used by Google for large scale products like Google Analytics and Google Earth and scales to thousands of machines and petabytes of data (Chang et al., 2008). Sensors often produce information which are repetitive in nature thus it is efficient to represent them in a highly structured manner rather than in a plain text (Swaroop and Shanker, 2010).

XML

The Extensible Markup Language (XML) is a simple and flexible data format and it is derived from the Standard Generalised Markup Language (SGML) (ISO8879:1986, 1986). It is widely used as data exchange and storage format on web and on sensor networks and is the accepted industry standard (W3C, 1996). A simple XML example has been presented at code listing 4.1 which consists of temperature sensor data. It has tags such as <sensorData>, <sensorId> and <unit> which are defined by the user as per requirement of the data being represented and is readable to both human and computers. It has also been used to communicate data from Internet of Things (IOT) to the web along with other data formats (Orestis et al., 2012). It has been used to share various data in a real-time decision support system for pandemic re-

sponse (Kelley et al., 2011). The Sensor Model Language (SensorML) which is widely used to describe process of measurement by sensors is also based on XML (OGC, 2007).

Listing 4.1 XML representation of sensor data

```
<?xml version="1.0" encoding="UTF-8"?>
<sensorData>
  <sensorID>phidgetspatial003</sensorID>
  <time>2012-09-12 04:38:36+00:00</time>
  <temperature>
    <unit>Celcius</unit>
    <value>25.4</value>
  </temperature>
</sensorData>
```

Listing 4.2 JSON representation of sensor data

```
{ "sensor_data" : { "sensor_id" : "phidgetspatial003",
                    "time" : "2012-09-12 04:38:36+00:00",
                    "temperature" :
                    { "unit" : "Celcius",
                      "value" : "25.4" }}}
```

Listing 4.3 Hexdump for compressed binary of XML data

047868d3933e5e9cf861e9a70ec60c19ca0a0239803c3cbb31e9ebcc0425a4

However XML is very verbose due to the repetitive usage of tags to represent elements and attributes in a precise tree structure as shown in code listing 4.1. Formats like JSON and YAML are increasingly becoming popular mainly because they represent data more concisely by using less repetitive tags. Code listing 4.2 shows the JSON equivalent representation of the sensor XML data above which is slightly concise than the XML counterpart.

Knowledge about the data types retrieved from the meta data or the schema allows efficient compression of XML (Moore et al., 2013). The compression makes XML competitive in terms of size to JSON and YAML. The XML schema is a very important aspect of this system as it can be used as a network and application protocol and also used for data validation and compression. A simple schema for the sensor XML data above is presented at code listing 4.4. It defines data types for the sensor XML such as *string*, *decimal* and *unix-time*. Further restrictions can be defined in the schema such as to limit the maximum number of characters in a string or to control the range of

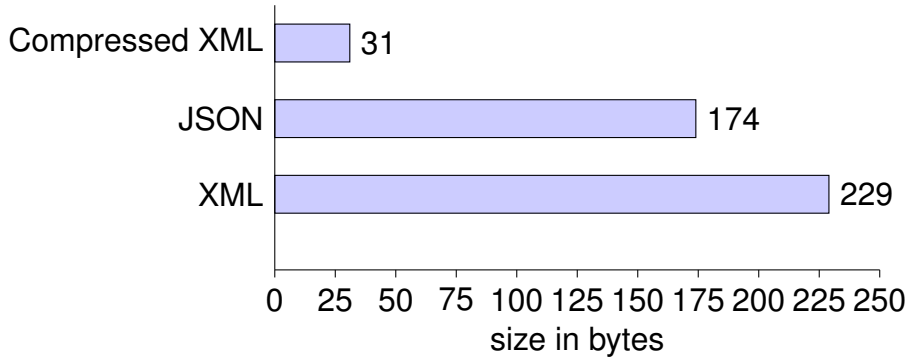

Fig. 4.1 Data size comparison

integers. The data type definitions and extra restrictions can be used to validate data defined in the XML and to efficiently optimise it. In order to compare different data representation sizes, the sensor XML data is compressed with the help of the schema using the Packedobjects library which is discussed in more detail in the next section. As shown in figure 4.1, the XML representation is bigger than the JSON one but the compressed XML is considerably smaller than the JSON representation. It means the verbose XML does not need to be sent over the network. The data can be decompressed later using the schema to retrieve the original data. The hex-dump for the compressed binary XML data is shown in code listing 4.3.

Listing 4.4 XML Schema for sensor XML data

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:include schemaLocation="packedobjectsDataTypes.xsd"/>
  <xs:element name="sensorData">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="sensorID" type="string"/>
        <xs:element name="time" type="unix-time"/>
        <xs:element name="temperature">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="unit" type="string"/>
              <xs:element name="value" type="decimal"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Veillard (1999) have developed a robust, highly portable XML library namely *libxml2* which provides an efficient XML parser for processing and searching data. There is huge advantage of using the parser as there is no need to convert data to another format in order to make it search-able. The data can be stored on local devices in XML format which removes the necessity to convert XML and maintain a relational database. Data format conversion can take a significant amount of time in real-time applications (Kulkarni et al., 2012).

4.2.2 Packedobjects

Packedobjects (PO) is an XML compression library which utilises XML Schema for both validation and compression (Moore, 2010). It performs efficient and fast encoding/decoding by utilising the knowledge derived from the schema. It uses a set of built-in data types to efficiently compress XML data by applying encoding rules which originated from ASN.1. Data validation is achieved along with compression by adhering to strict encoding which enables lossless decoding and a better compression ratio. It can operate in constrained hardware and limited network resources too (Moore et al., 2013).

Although the concept of XML compression is not new, it can be improved by applying semantic knowledge of the data types with in the current context. Traditional text compression methods do not consider semantic knowledge and XML compression tools such as EXI use schema but do not consider the current context of the data. An IPv4 address such as the loopback device address "127.0.0.1" is described in dot-decimal notation and is considered a string of 9 characters by a text compressor (Moore et al., 2014b). However context knowledge can be used to imply that all IPv4 addresses are 32 bits or 4 bytes in length. Thus the extra semantic knowledge can be used to improve the XML compression. Similarly UNIX timestamp data such as "2014-07-15 20:28:27Z" can be encoded to 4 bytes instead of a string of 19 bytes by representing them as the time since an epoch.

PO achieves efficient compression by packing the data at bit level instead of byte level. Bit level compression may not be suitable for systems which require very high speed performance due to the extra processing required to achieve bit stuffing. However PO still performs better or similar with compare to the most commonly used XML compression tools such as EXI and XMILL and text compression tool such as zlib (Kheirkhahzadeh et al., 2013; Moore et al., 2014b). For example data representing option or choice can be effi-

ciently encoded using context information with the help of schema. The information about the type of sensor such as accelerometer, gyroscope, compass or GPS can be stored a string of characters. The type of sensors available for use is limited and thus can be represented as enumeration. A list of 4 different types of sensors can be represented with only 2 bits as shown below. Similarly a list of 128 options can be represented with just 7 bits by defining the enumeration in the schema in advance.

1. Accelerometer as 00
2. Gyroscope as 01
3. Compass as 10
4. GPS as 11

4.2.3 Publish/Subscribe messaging

Structured data representation makes it possible to communicate selected data based on its features rather than the network address of a sensor node and this approach is commonly known as data centric communication (Estrin et al., 1999). Publish/Subscribe (Pub/Sub) messaging systems are common examples of such communication and have been widely used to provide topic based communication. However mobile networks have different challenges compared with traditional networks. To solve these new challenges, Pub/Sub messaging systems are emerging which provide greater scalability to the network by decoupling the data senders and receivers from each other. They are considered more suitable for the networks with dynamic topologies such as sensor and mobile networks (Eugster et al., 2003). The data can be filtered at the time of the publication based on the interests of the active subscribers. Pub/Sub systems offer efficient continuous processing and filtering of data in distributed and heterogeneous environments and have been used to create energy efficient mobile crowd-sensing platform (Podnar Zarko et al., 2013).

A simple Pub/Sub system as shown in figure 4.2 consists of a single publisher to broadcast data and multiple subscribers to receive them (Eugster et al., 2003). The communication is normally one way which means subscribers do not acknowledge to publishers which makes it simpler to implement and faster for large scale networks. However these systems are only

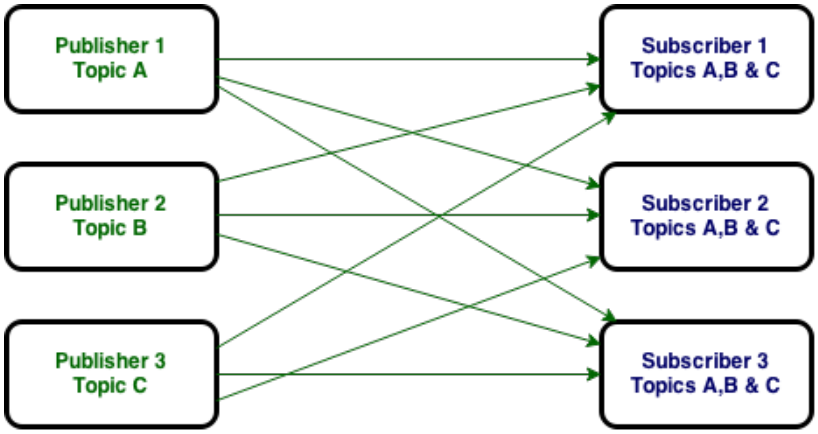

Fig. 4.2 Simple Pub/Sub system

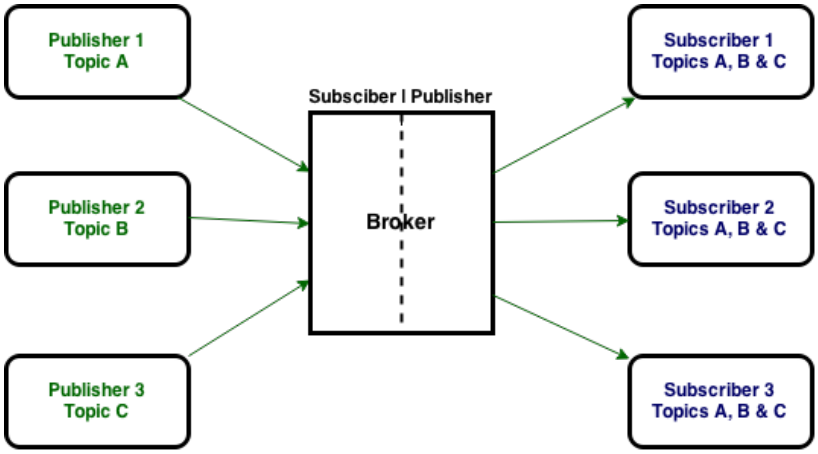

Fig. 4.3 Broker based Pub/Sub system

suitable if the publisher has a permanent network address as it becomes difficult for subscribers to follow publishers without a permanent address. It becomes even more difficult for subscribers to manage multiple connections if they receive data from many publishers at the same time. However an intermediate device called as broker can be introduced to solve these issues where subscribers communicate to the publishers via them (Hunkeler et al., 2008). The broker acts as a publisher to a subscriber and as a subscriber to a publisher and hence facilitate the end to end information delivery shown in figure 4.3. The publishers and subscribers require only the network address of a single broker instead of multiple publishers or subscribers which makes it simpler to manage connections.

It is simpler to implement and manage broker based Pub/Sub systems.

Equation 4.1 shows the total number of connections required for simple Pub-/Sub model. And equation 4.2 shows the total number of connections for a broker based model. For example, broker-less Pub/Sub model requires 9 connections to connect 3 publishers to 3 subscribers. On the other hand only 6 connections can serve the same purpose after using an intermediate broker. Similarly 60 connections can connect 30 publishers to 30 subscribers with broker in the middle where as 900 connections are required to do the same otherwise. The number of connections required for the simple model grows rapidly faster with the increase in the number of publishers and subscribers as shown in figure 4.4 and hence becomes difficult to manage quickly.

$$(Pub_m * Sub_n) \tag{4.1}$$

$$(Pub_m + Sub_n) \tag{4.2}$$

Where Pub_m is the number of publishers and Sub_n is the number of subscribers.

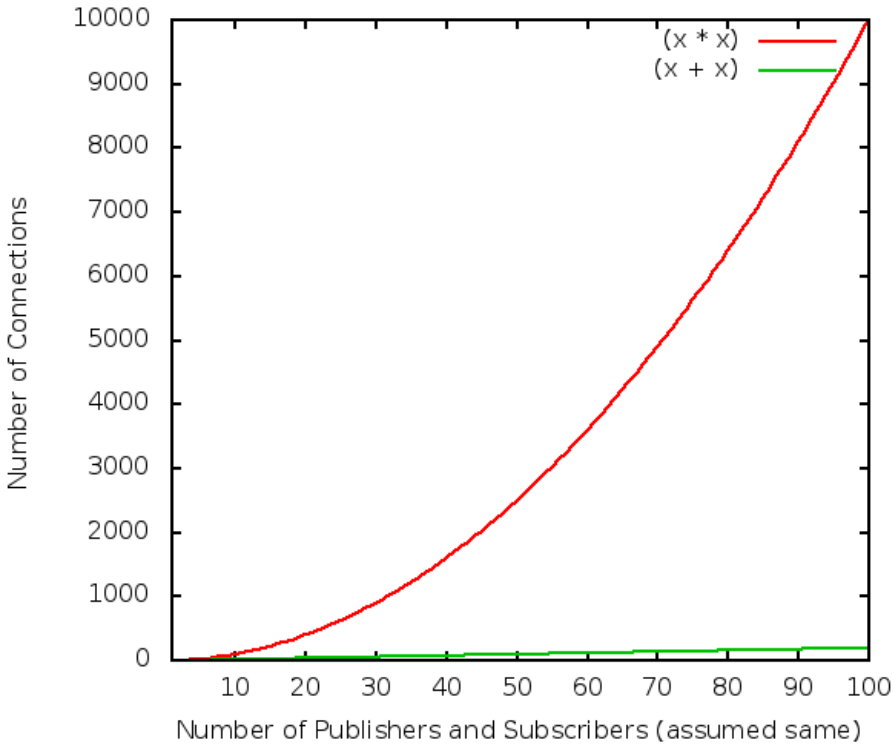

Fig. 4.4 Total connections comparison

The broker does not store any data permanently as it only forwards data from publishers to the connected subscribers. The data is mainly stored at the publisher side where it is generated. The subscribers can retrieve it using the Pub/Sub messaging model via the broker. The aim is to enable real-time access to the data at its origin in a simple/scalable way and thus avoiding the necessity to transfer the data to a central server before being available to the subscribers.

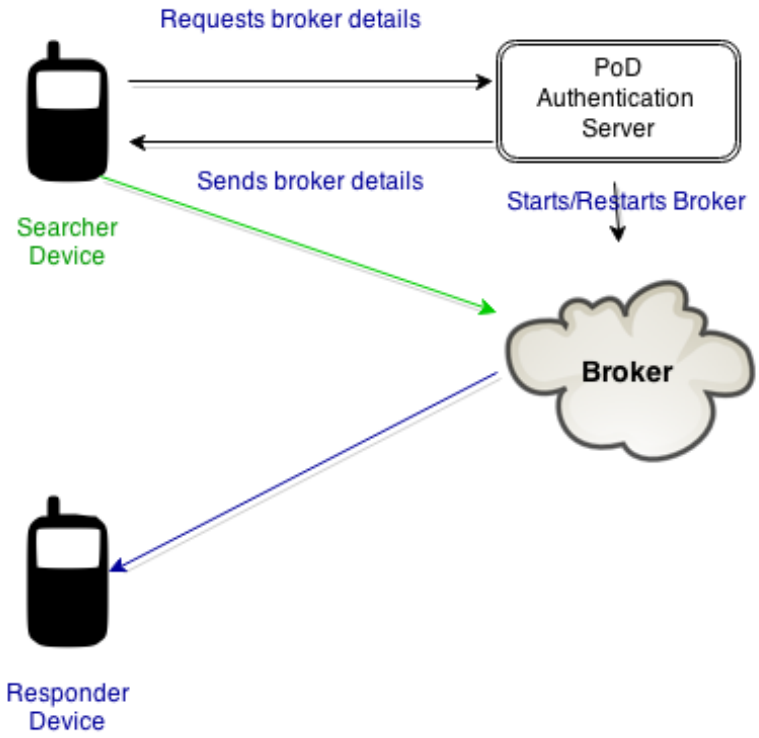

Fig. 4.5 PoD authentication server

The broker however could suffer from the risk of single point of failure. In order to avoid this failure, an extra authentication server is added to the PoD system as shown in figure 4.5. The system thus maintains an authentication server to initialise the publishers and subscribers where it provides the broker details to them. The nodes initialise to the server by using the XML schema for the data they wish to communicate. The server then starts the respective broker if it's not already running so that the nodes can start communication via them. The nodes can also contact the authentication server in case of a broker failure which can restart the concerned broker immediately. The authentication server can also monitor the brokers continuously in order to reduce the downtime further.

ZeroMQ

Various messaging protocols/libraries implement a Pub/Sub messaging pattern to provide asynchronous communication (Amir et al., 2004; Hintjens, 2007; Hunkeler et al., 2008; RabbitMQ, 2011; Shevat, 2004; Snyder et al., 2010). ActiveMQ, MantaRay and RabbitMQ messaging systems provide robust and powerful Pub/Sub messaging but are more suited to large enterprise environments. Those systems are less suitable for mobile devices running on constrained resources and bandwidth. Bagale et al. (2012) have studied ZeroMQ and Spread messaging systems focusing on their suitability for constrained devices as discussed in previous chapter. ZeroMQ was found to use less control overhead per message and was simpler to implement. The messaging framework used in the system is built using the ZeroMQ library which is defined as the socket library which works as a concurrency framework (Hintjens, 2007). It is a light-weight, fast and scalable messaging library which has been already used to create a range of real-time applications including stock trading, network traffic monitoring and electricity monitoring applications (Hintjens, 2010). It has been already ported to all major platforms and consists of bindings for most programming languages. Dworak et al. (2011) found ZeroMQ to be the best middleware messaging system during a study of different middleware platforms for use in the *Large Hadron Collider* project. ZeroMQ has also been notably used by NASA, Spotify, Microsoft and Zynga for their messaging requirements (Hintjens, 2012).

ZeroMQ provides some extra features for Pub/Sub systems to improve the overall performance. Most Pub/Sub systems send messages for all the topics to a subscriber which then filters them on its side based on its own interest. Thus messages are sent over the network even when it's not necessary. It may not be an issue for systems with unlimited bandwidth but it can affect the performance of systems with limited bandwidth. However ZeroMQ performs topic filtering at the publisher side instead of the subscriber side. Thus a subscriber only receives the messages with the topics it has subscribed. The ZeroMQ broker also forwards subscription messages to publishers allowing them to drop messages when there are no connected subscribers. Thus publishers can broadcast messages for topics which have active subscribers. Message queue size can be defined in advance on both publisher and subscriber side to isolate a slow subscriber problem and stop it from affecting the whole system.

4.3 PackedobjectsD architecture

The core architecture of the data sharing system consists of an existing message format, messaging architecture and compression technique. It combines an XML data format, XML schema, Pub/Sub messaging model implemented with ZeroMQ library and the PO XML compression library in order to provide a light-weight scalable data sharing system as shown in figure 4.6. The overall architecture is called PackedobjectsD (POD) inspired by the *Packedobjects* compression library and *D* stands for distributed. Applications need to obtain the XML schema before participating in the communication and it has to be done outside the POD framework. It can be compared with the exchange of a secret key before communicating using symmetric encryption. The schema is used to compress and decompress the XML data during the communication process.

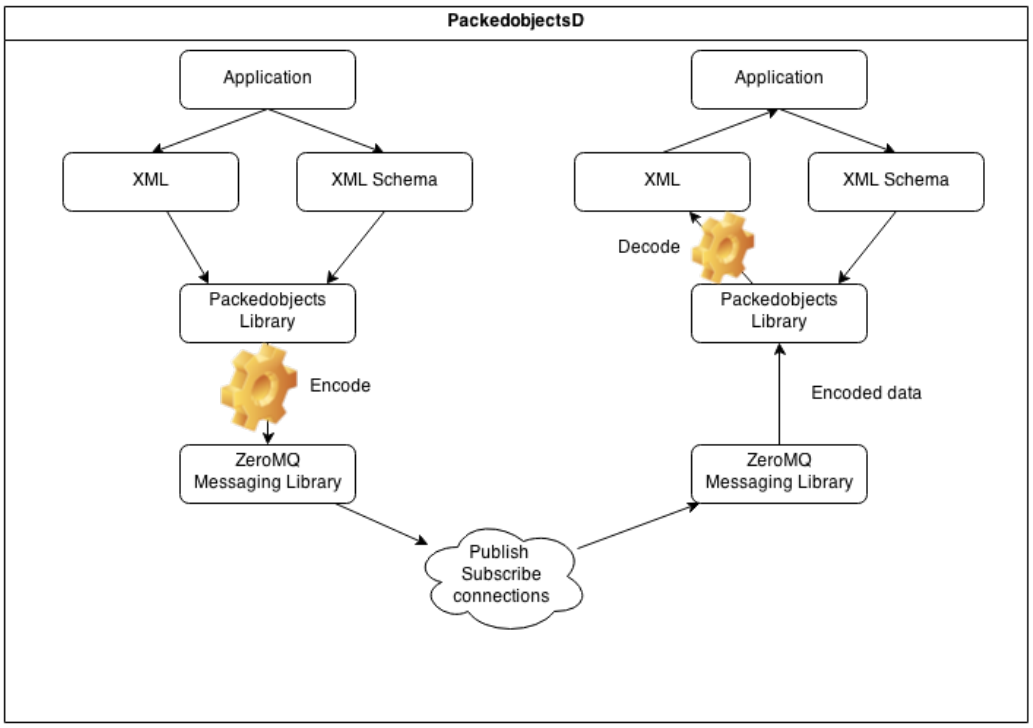

Fig. 4.6 Architecture

Applications process the data in XML format which is validated and compressed using an XML schema. The actual compression of XML data is performed by the PO library using knowledge derived from the schema. Once the data is validated and compressed, it is passed on to the ZeroMQ messaging framework which broadcasts the data to all connected applications

which have subscribed to receive that information. The messaging framework broadcasts the data via an intermediate broker as discussed above. The ZeroMQ messaging framework on the receiving application passes the received compressed data to the PO library which decompresses and validates it using the schema it has obtained before the communication. The XML data is then be passed to the application for further consumption and/or storage. The schema is an integral part of the data sharing system as it is used for various purposes besides being used for XML compression.

4.3.1 XML Schema

The schema is used for four different purposes.

1. Network Protocol
2. Group identity
3. Data compression
4. Data validation

It is used as a **network protocol** for the system as it defines what kind of data is being communicated. The applications can agree on what to communicate by using a common schema. It is also used as the **broker's identity** to group application nodes of similar interests together. The nodes who communicate using the same schema belong to the same group and can share data to each other.

It is also used to **validate data** before and after communication. The schema can define restrictions on data e.g. range of integers, length of strings or choice of values which can be used for data validation. Validation is an integral part of any data sharing system and thus combining it with the compression process is beneficial for saving time. And it is mainly used to efficiently **compress the data**. Although the concept of XML compression is not new, it has mostly been performed without the schema as text compression. Even the schema aware compression techniques do not consider the semantics or the context of the information while compressing. As explained above in section 4.2.2, schema aware XML compression combined with context aware data types for information can be used to efficiently compress the XML data and performs similar or better than commonly used tools such as EXI, XMILL or zlib (Kheirkhahzadeh et al., 2013; Moore et al., 2013, 2014b).

4.3.2 Mobile based product search system

A mobile product search system has been built using the POD architecture in order to evaluate its performance. The data is communicated in both compressed and uncompressed format to find out possible benefits and trade-offs of compression on data sharing systems. The aim is to find out if the time required for achieving the data size reduction from compression is not considerably longer for it to be useful. There is no advantage of performing the compression if it takes more than the actual data communication.

The system comprises of various searching and responding nodes which are grouped together based on a predefined XML schema. The responder nodes maintain records about various products such as laptops, tablets or any other everyday item along with their location information. The records are stored in XML format and conform to data restrictions as defined in the schema. The node initialises with a schema, node type (searcher or responder) and other basic information to the POD system which then sends back the details about the broker. A searcher node then broadcasts the search queries to the responders nodes via the broker within the POD system as shown in figure 4.7. Responders process the search query and look up on their database for matches and reply back with matching results if any as shown in figure 4.8. The searchers are able to search data on its origin in real-time. For example, a responder may be moving from one location to another and the searcher can search for products located in a certain area instantly. It is not necessary for a central server to update its index with latest location information about the contents of the nodes in order to provide correct result in real-time.

The nodes act as both publisher and subscriber in order to complete data sharing. A searcher node publishes search queries to the subscribed responders and waits to receive the response back from them. Then the responder nodes publish the search results back to the searcher based on the matches on its record. However, the actual implementation is simple as the complexity of network connections is hidden from the nodes by the POD architecture and the nodes can simply join the group as a searcher or a responder and start communicating with the group. In addition, the broker within the architecture facilitates communication for both scenarios so that nodes can forget about the complexity of other connected nodes. The POD architecture provides simple interface to them by hiding the broker implementation and other network

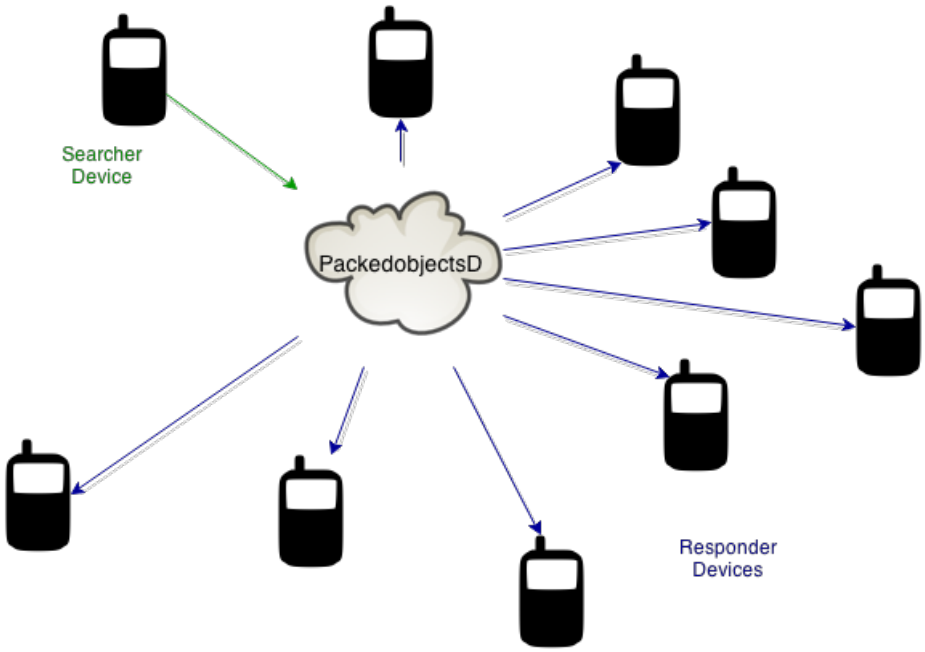

Fig. 4.7 Searcher nodes

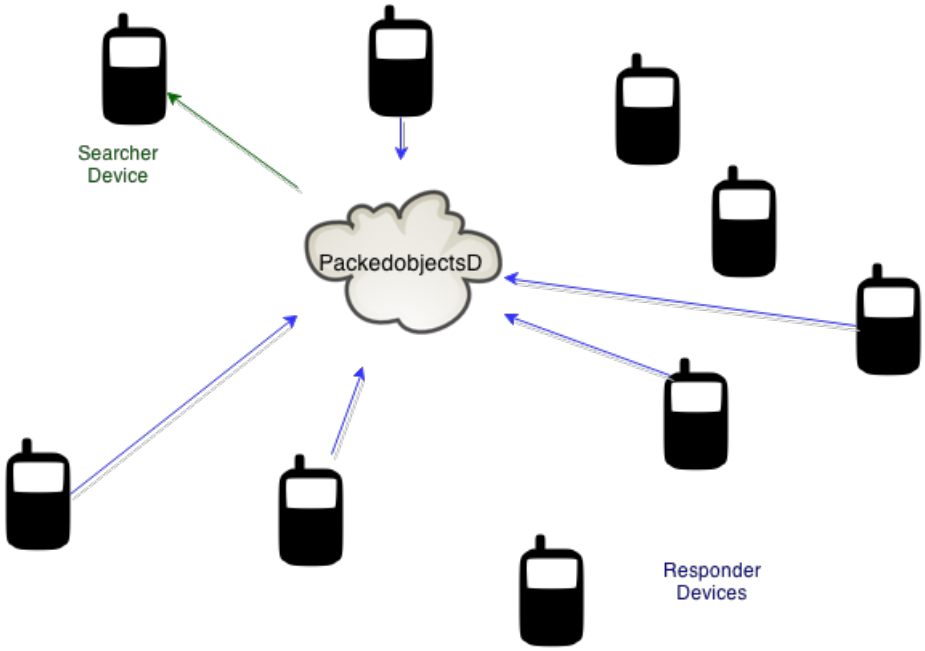

Fig. 4.8 Responder nodes

infrastructure details from the nodes. The broker is currently running on a server but it does not have any specific requirement for that and can be run on any mobile device without affecting the system. The device however must have a permanent network address and must be able to run continuously.

4.4 Experimental setup

The experiment was conducted using a mobile application implemented on Blackberry Z10 mobile phones which run UNIX based Blackberry 10 OS. The application is based on buying/selling everyday items. The main aim of the experiment was to evaluate the difference between communicating plain and compressed XML data. The XML compression time and network transfer time for compressed data is measured along with the time for communicating the uncompressed XML data. However the plain XML data must be serialised or converted to some string format before it can be passed on to network. The compression is done using PO library and plain XML serialisation is done using *Libxml2* library. The data compression and decompression time for both approaches is measured using the *clock()* defined in GNU C library (GNU, 1999). It measures the CPU tick counts while the compression/decompression process is running to provide the elapsed time. The compression was repeated 3000 times and the average value is obtained in order to remove one time and system level effects. The experiment is also repeated multiple times (17 times) with different input data to remove similar effects. The X-axis labels 1-17 in figures 4.9 to 4.15 represent different input data sets. The main concern about the PO compression is to find out if it takes considerably longer than the plain XML serialisation process in order to justify the benefits it provides on data size.

4.5 Results

Figure 4.9 shows the size of original XML data and size after PO compression. The XML data has been compressed on average from 145.5 bytes to 11.5 bytes which is merely 8% of the original size. Although the actual compression ratio may vary slightly, it is better than most text compression tools Moore et al. (2014b). Figures 4.10 and 4.11 show the CPU time elapsed while compressing and serialising the XML data respectively. The PO compression process encodes the XML data to binary data which is also serialised for network communication and also performs data validation checks. Hence the actual CPU time elapsed includes data validation, compression and serialisation. The XML serialisation process however does not perform data validation and compression as it only serialises XML data for network communica-

tion. The XML serialisation is thus obviously faster than the PO compression process as it performs fewer tasks. The average PO compression time is 128 microseconds while average XML serialisation is only 15 microseconds. However the time difference for those two approaches is negligible and does not affect the overall communication time. Also data validation needs to be performed separately when XML is serialised using Libxml2 library requiring extra time.

$$time(s) = datasize(bits)/networkspeed(bit/s) \tag{4.3}$$

Although it takes less time to serialise plain XML, extra amount of data needs to be communicated when it is not compressed. As shown in figure 4.9 134 extra bytes are sent over the network per 145.5 bytes. The network packet overhead is higher when communicating plain XML data with compare to PO compressed data. Transmission latency and re-transmissions from packet losses increases the overall network communication time further. The overhead, latency and packet loss can be expected to affect PO compressed data much less as only 8% of the original data is being sent over the network.

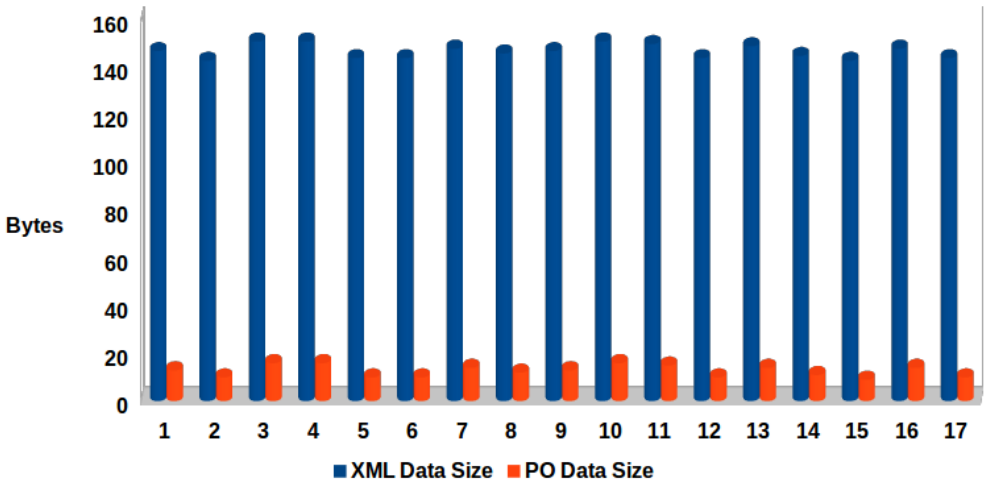

Fig. 4.9 Encode data size comparison

In order to estimate the overall communication time involving compressed and plain data, the theoretical data transfer time for both types of data is calculated using equation 4.3. It is assumed that data transfer time for 3G networks is 1Mbps for calculating the approximate data transfer time. So that the overall time for communicating compressed data including compression time can be

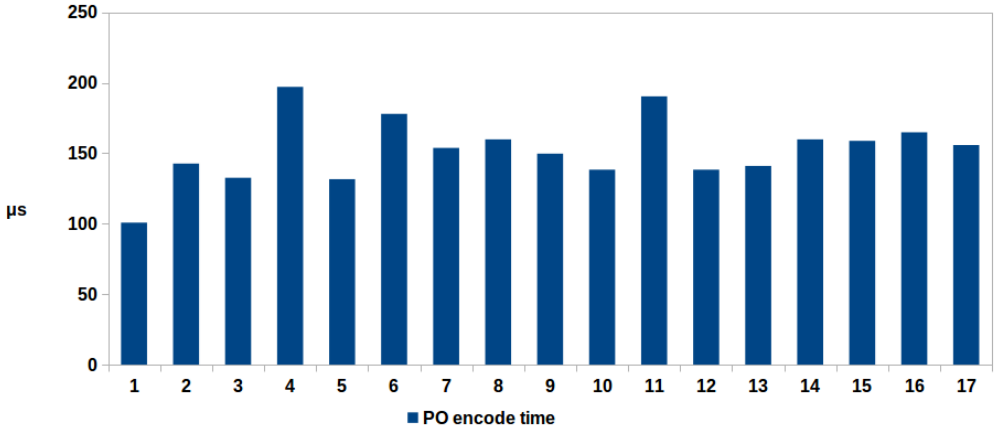

Fig. 4.10 PO encode CPU time

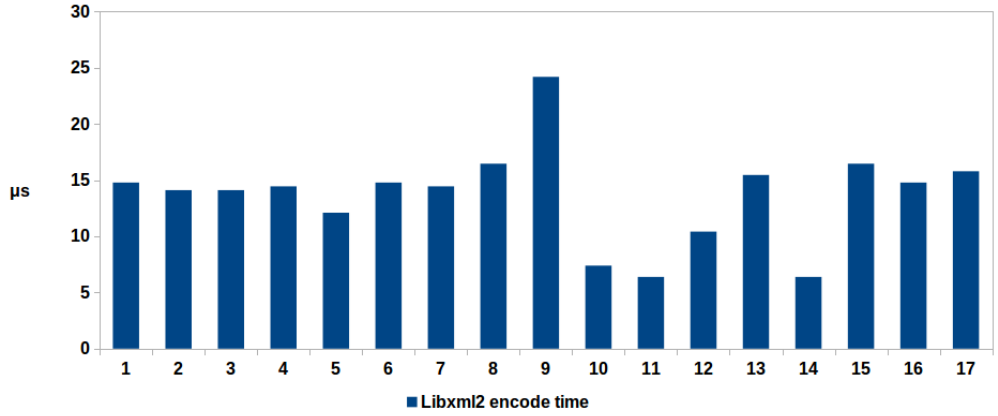

Fig. 4.11 Libxml2 encode CPU time

compared against the time for communicating plain data including serialisation time. The data transfer time for other network bandwidth speeds is shown in figure 4.16 later. Figure 4.12 shows the combined time for PO compression and data transfer and the average is 140 microseconds. Similarly figure 4.13 shows the combined time for XML to string serialisation and data transfer time and the average is 164 microseconds. The extra time spent while validating and compressing the data is compensated during data transfer. It takes less time to serialise uncompressed XML but more time to communicate it. It takes less time on the overall to communicate the PO compressed data.

Similarly the decompression/deserialisation time for both approaches is also measured using the same XML data. A responder was receiving the search query explained above and figures 4.14 and 4.15 show the CPU time for respective processes. The PO decompression process includes the data de-serialise, decode and validation while XML de-serialisation process only

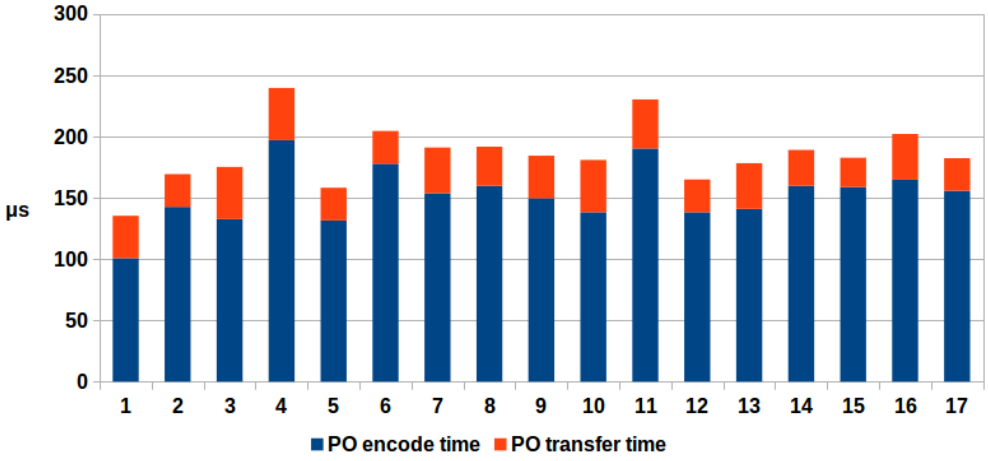

Fig. 4.12 PO encode and data transfer time

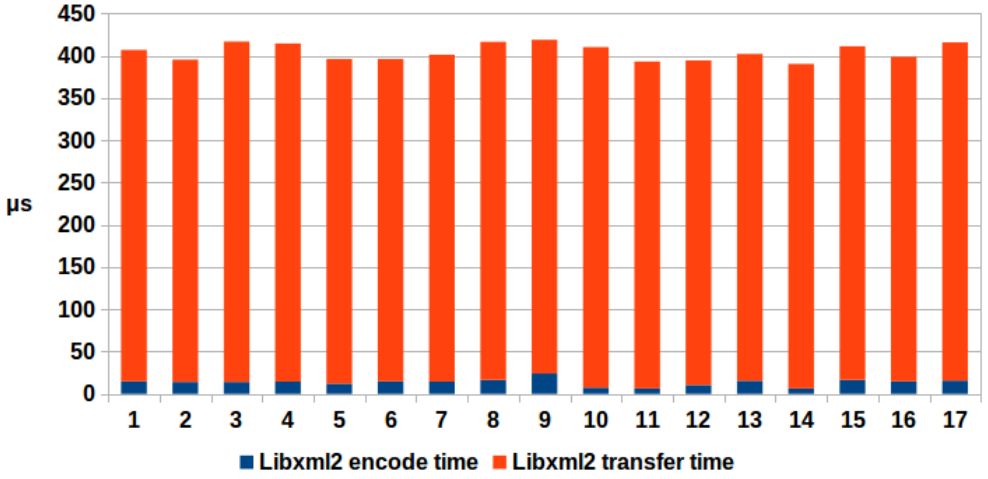

Fig. 4.13 Libxml2 encode and data transfer time

performs data de-serialising. The average CPU time for PO process is 257 microseconds whereas the CPU time for XML process is 399 microseconds. The PO decompression process is faster with compare to the XML one.

4.6 Discussions

In order to evaluate the overall impact of PO compression on the networks of different bandwidth speed, the theoretical data transfer time for the **average** PO compressed data and uncompressed XML data is calculated using equation 4.3. The CPU time for the compression and serialisation has been ignored during this calculation as it remains constant on the device regardless of the difference in bandwidth. The theoretical data transfer time is calculated

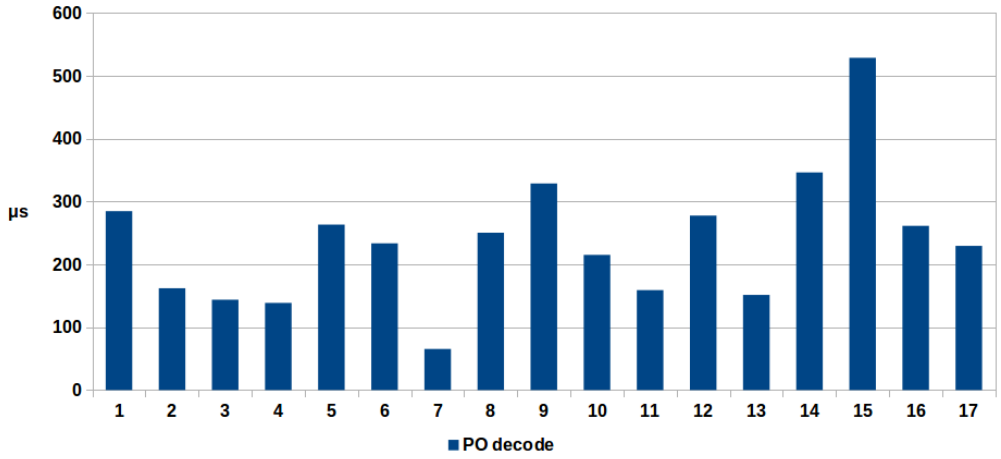

Fig. 4.14 PO Decode CPU time

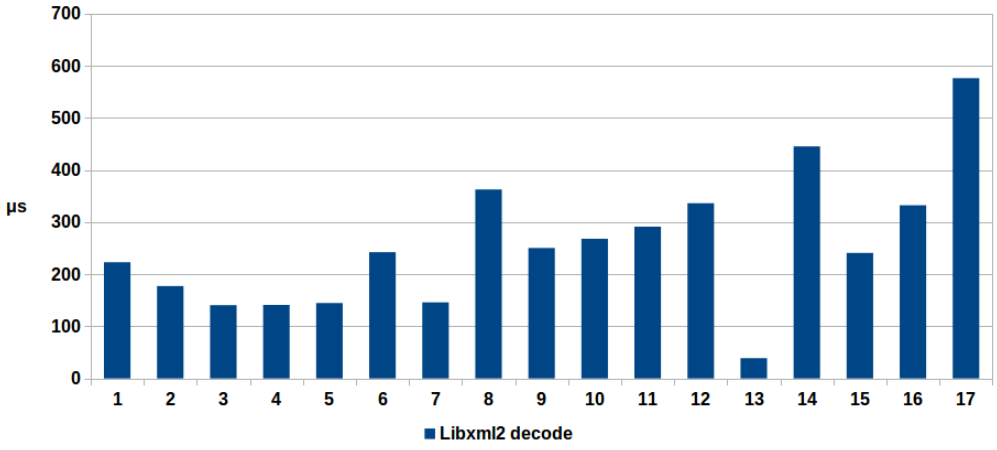

Fig. 4.15 Libxml2 Decode CPU time

for various network bandwidth speeds ranging from 150Kbps to 4Mbps covering low bandwidth sensor networks and mobile networks such as GPRS, 2G and 3G. As shown in figure 4.16 the transfer time difference is less for faster networks and it increases rapidly for slower networks. Thus PO compression is less beneficial for networks with high bandwidth but it is significantly efficient for low bandwidth networks.

The amount of data communicated within a group can become unmanageable quickly for the mobile network with the increase in number of nodes. For example, if a node is receiving 1000 bytes of data per second from 1000 nodes, it receives around 0.95MB of data every second. The data size may not be significant for traditional networks but most mobile networks have typical bandwidth in the range of 1-2 Mbps. But there are various bandwidth hungry applications such as online games, location based services and video

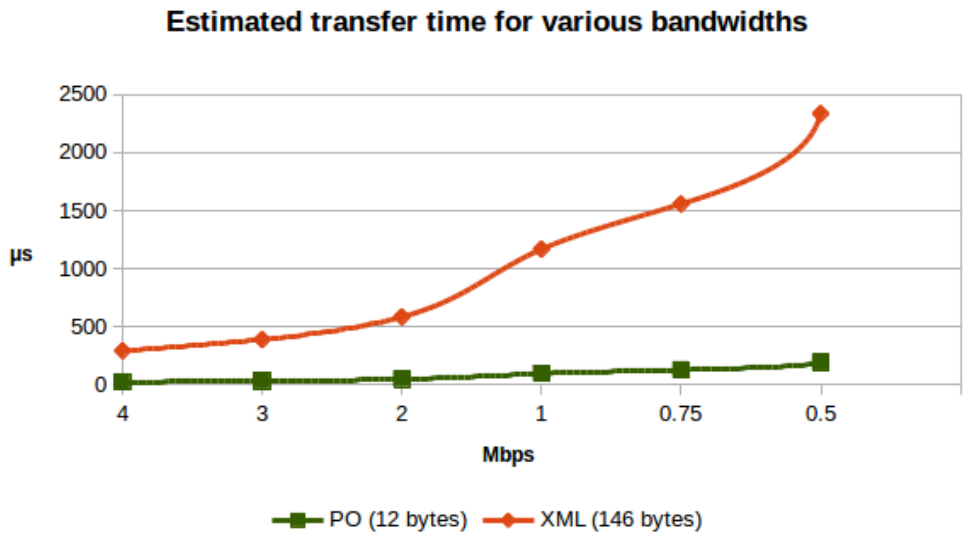

Fig. 4.16 Transfer time comparison for various network bandwidths

streaming which consume a lot of available bandwidth. It is not ideal for a single application to consume such a high ratio of available bandwidth and scalability can become issue very quickly. However the data can be reduced up to 0.08MB in size by using PO XML compression as discussed above. Also it is very important for an application which runs continuously in background, to consume as little bandwidth as possible to be considered useful by users.

However there is a possibility of the broker being the bottleneck of the overall network as the data always passes through it. The worst case scenario is when all the nodes in a network send data to all the remaining nodes at the same time. The ZeroMQ broker implementation encapsulates multiple network connections inside a single subscriber socket. For example if 1000 nodes as explained above are subscribed to receive data from all other remaining 999 nodes via the broker, there is only a single socket connection between the broker and any subscriber instead of 999 separate connections. The data is then fair-queued on the socket by the broker for the subscribers to receive. Thus, the broker only receive maximum of 1000 bytes each from 1000 publisher nodes in one second and it does not need to receive the same data more than once from a publisher despite 999 subscribers receiving it. It is an important feature of the broker and helps to reduce bottleneck by avoiding duplicate transmissions from the publisher. On the other hand the broker remains under more pressure to forward those messages to the subscriber nodes. For the above explained scenario, each subscriber is now expected

to receive 0.95MB of data and the broker must be able to forward these messages fast enough to avoid network bottleneck. This is a possible problem with the broker based architecture for the data sharing system.

So let's look at an alternative broker-less architecture to see if it can improve the overall network congestion. It is possible to connect all 1000 nodes to each other by having direct connections between them by ignoring the complexity. However each publisher must send its copy of 1000 bytes of data every second separately to each of the 999 subscribers. Thus all the 1000 publishers are sending 0.95MB of data per second causing the overall network to carry exactly same amount of data as the broker based approach. The problem of single point bottleneck is solved however the network congestion still remains the same. Thus broker-less approach is also not a good solution considering the complexity of huge number of connections and the dynamic discovery problem. The broker however could be running on a faster network with compare to the rest of the nodes allowing it to forward the messages faster to avoid network bottleneck.

The system uses XML data format for storage as well as communication. XML data is directly searchable so there is no need for converting it to any other data format such as SQL. The time required for data type conversion is considered to be significant if a query is being made over a network (Kulkarni et al., 2012) and helps to improve the overall speed of the communication by avoiding frequent data conversion.

4.7 Summary

In this chapter, a scalable real-time data sharing system suitable for mobile devices is discussed for collecting, storing and communicating sensor and user data. The messaging infrastructure is light-weight considering the constrained resources available in mobile devices. The Pub/Sub message model implemented using an intermediate broker enhances scalability as nodes are able join/leave easily without worrying about the dynamic discovery problem of other nodes. The number of end to end connections required is significantly small especially when the network is big. The sensor and user data is processed in XML and its schema is used for utilising the strengths of the structured format. The schema is used for various purposes such as data optimisation and validation, group identity and as network protocol. Schema

allows compressing data more efficiently than well known text compression tools such as gzip which helps to create a scalable and light-weight real-time data sharing system. The PO compression and decompression process takes less time with compare to plain XML serialisation and de-serialisation process while providing extra data validation and efficient compression. The compression significantly improves overall performance of low bandwidth networks but can still be used for high bandwidth mobile networks such as 3G to improve the performance.

Chapter 5

Energy consumption trade-offs for schema-aware XML compression

The development of sensor and embedded hardware technologies coupled with the reduction in their price means embedded devices are able to provide a range of versatile services. Wireless data transmission over sensor networks is known to consume a significant share of energy compared to data computation on the device itself. Thus data compression techniques have been used to reduce the amount of data to be communicated over a network at the cost of extra processing time on the device. This chapter investigates the energy consumption trade-off of XML compression and data communication on embedded devices. Various experiments are conducted to analyse schema-aware XML compression that can significantly reduce overall energy consumption with a focus on energy consumption of network communication of compressed and uncompressed data.

5.1 Background

The evolution of sensor technologies has allowed embedded devices to collect a range of information about a user's location, activity or their environment which is then be communicated over the network. It is fundamental for data sharing systems to consider the impact on battery consumption. Data compression can be used to reduce the number of bits communicated over network and to minimise overall energy consumption. The aim is to investigate this further by analysing the impact of schema-aware compression on energy consumption and compare it with other compression techniques. The Exten-

sible Markup Language (XML) is a simple and flexible data format derived from the Standard Generalised Markup Language (SGML). It is widely used as a data exchange and storage format on the web and sensor networks and is the accepted industry standard (ISO8879:1986, 1986; W3C, 1996). However it is very verbose because of the repetitive usage of tags which represent the data structure. As a result of this, alternative formats such as JavaScript Object Notation (JSON) are considered for their conciseness. However knowledge about data types can be used to efficiently compress XML data to make it competitive in terms of size (Moore et al., 2014b). Although the schema-aware compression is very efficient and the compression ratio for typical XML data is more than 80%, the gain must also be justified by the energy cost of the compression process. If it requires more energy to compress the data then to communicate the uncompressed data, the compression process cannot be considered beneficial in terms of energy.

5.2 Related Work

Embedded devices often have limited resources such that energy consumption can affect the life of the device on the sensor network. Thus it is important to establish strategies to reduce energy consumption. Lossy compression techniques have been used to save energy consumption at the expense of reduced audio and video qualities (Sinha et al., 2000). However this thesis focuses only on reducing energy consumption without sacrificing the quality of data to be transmitted. The energy cost of wireless communication is considered to be significantly higher than on device computation. It requires over 1000 times more energy to transmit a single bit of data than a single 32-bit computation (Barr and Asanovic, 2006). Thus, it may be efficient in terms of energy to perform 1000 computations in order to compress the data by 1 bit. Earlier work has analysed different compression algorithms to study their impact on energy consumption mainly focusing on the popular implementations of Lempel-Ziv 77 (LZ77), Burrows-Wheeler Transform (BWT) and Lempel–Ziv–Welch (LZW) algorithms (Barr and Asanovic, 2006; Wang and Manner, 2009; Xu et al., 2003).

In (Xu et al., 2003), *gzip* compression software (based on LZ77) was found to be superior to *bzip2* (based on BWT) and *compress* (based on LZW) in terms of energy saving in a hand-held device using a variety of file types. The

analysis was made while data, compressed in advance using these three algorithms, was downloaded from a server and decompressed on the hand-held device. The energy savings from the download time and decompression process was used to find the most energy efficient algorithm. Thus the algorithm with the best compression ratio was not necessarily the most efficient in terms of energy savings despite having smaller a download time (and less energy spent during download) due to the increased energy cost in decompression. Although the results are beneficial for applications which primarily download information, it cannot be applied to scenarios which may upload compressed information. In (Barr and Asanovic, 2006), energy consumption of algorithms *bzip2*, *compress*, *ppmd* (based on Prediction with Partial Match), *lzo* and *zlib* (both based on LZ77) are evaluated on a hand-held device and the algorithms focused on compression ratio and execution speed. Uncompressed data was also copied to and from the network to measure energy consumption without compression. The experiments were done with 1MB of compressible web and text data and were repeated with slightly different optimisation parameters for most of the algorithms and separate measurements were made for energy consumption by CPU, memory, network and peripherals. Most compressors perform well and save energy compared to when uncompressed data is communicated. Compressors *lzo*, *compress* and *zlib* (with effort level 1) save almost 50% energy overall which includes compressing & sending and receiving & decompressing the data compared to uncompressed data communication. The bad performance of some compressors is mostly related to the extra time spent on CPU and memory for compression. Additionally, the decompression process was found to cost less energy than compression.

Similarly Wang and Manner (2009) present the results of experiments involving mobile devices while communicating various types of web content. The results are similar to previous experiments as *gzip* and *lzo* compressors are concluded to be the most efficient energy savers. In (Gil and Trezentos, 2011), XML is compared with JSON and protocol buffers for compression energy efficiency in smart phones. However only text-based compression with *gzip* is performed and XML schema is not considered at all. The energy consumption of all three data formats was almost similar for both small and large volumes of data. It is beneficial to perform compression on text formats in order to save energy consumption overall, however, binary formats do not provide energy savings when compressed due to longer time spent during compression. Although *Protocol Buffers* provides efficient data repre-

sentation in terms of size, the advantage disappears when XML and JSON are compressed. Also it cannot be used in dynamic environments as it requires recompilation of its schema and the application whenever the schema changes which makes it unsuitable for embedded devices where redeployment can be difficult. Also the energy measurements were made using software running on the mobile device and thus the accuracy may not be same as that of a hardware measurement. Similarly authors in (Szalapski et al., 2012) compared various XML compression techniques including gzip, XMILL and PAQ to Tinypack. The experiments found that the total time for sending compressed data is always lower than sending uncompressed data. PAQ reduces the data size most but it uses huge amount of memory and spends more time to process than to send uncompressed data

Earlier results imply that if it requires too much time or too much memory to compress data, it may not be beneficial overall in terms of energy despite significantly reducing the data. Thus compression must be performed to balance the trade-off between compression ratio, compression speed and CPU and memory consumption in order to provide overall energy consumption savings. Earlier work has considered schema-aware XML compression to achieve compression ratio better than text-based XML compression using simple and efficient compression techniques (Kheirkhahzadeh et al., 2013; Moore et al., 2013, 2014b). In this chapter, it is investigated further and the energy consumption of schema-aware compression techniques is evaluated to justify its advantages over text-based compression techniques or when there is no compression at all.

5.3 Experimental setup

In this section, the devices used during the experiments are discussed. The features of the device used to measure energy consumption of the compression process is also presented. The XML dataset used for compression and communication is explained focusing on the types of data in each dataset. The compression techniques used for compression are also briefly discussed. The formula to calculate the theoretical network transfer time for compressed and uncompressed data is also shown.

5.3.1 Device and Tools

The embedded device used for the experiment is a Raspberry Pi single board computer (SBC) which is running Raspbian OS and has 700MHz ARM processor and 512MB RAM (Pi, 2012). Energy consumption was measured using an ODROID smart power device which has output of 3-5.25V DC and can measure current or watts consumed by the connected device (Odroid, 2014). The energy consumed in Joules can be calculated using equation 5.1. The typical tolerance rate is 2% with a sampling rate of 10Hz. In order to measure energy consumption, the device is connected to the power supply of the smart power device as shown in figure 5.1 which measures the electrical current drawn.

$$Energy_{(Joules)} = Power_{(Watts)} * Time_{(seconds)} \tag{5.1}$$

Fig. 5.1 Odroid smart power measurement tool

5.3.2 Methodology

In these experiments, XML compression techniques Packedobjects (PO), EXI, ZLIB and Libxml2 are compared. Tinypack was considered for comparison as well but its implementation and the test XML data-sets weren't publicly available at the time of this research. EXIProcessor is used as the EXI implementation in the experiments (Garrett, 2012) and is referred to as EXI for simplicity in the coming sections. Similarly, ZLIB's example implementation created by

Table 5.1 Test XML files, data types and compression ratio

Name	Size(Byte)	Data Types	po	exi	zlib
ping	72	enumerated	72	24	1.09
timestamps	246	unix-time	16.4	2.51	1.7
router-addnet	510	string	5.48	5.31	2.16
sensor	549	numeric	21.96	17.16	2.53
iptel-ethinfo	627	numeric	17.42	16.08	2.2
iptel-devinfo	639	string	7.26	6.09	1.97
switch-config	678	string & ipv4-address	8.17	7.98	2.65
purchaseorder	738	string & numeric	7.38	6.31	2.31
router-disc	767	string & numeric	8.16	9.13	2.46
router-qos	817	string	6.92	5.79	3.13
personnel	856	string	11.41	9.41	2.89
menu	860	enumerated & string	78.18	47.78	2.88
unsafenumbers	1,155	numeric	15.61	16.04	5.04
temper-sens	1,307	string & numeric	16.97	18.41	3.72
vehicles	5,142	string & numeric	13.64	14.4	4.55

its original authors is used during the experiments (Gailly and Adler, 1995). PO and EXI perform schema-aware XML compression where as ZLIB performs text-based XML compression. Libxml2 is used to serialise XML without compression to make it suitable for network transfer. The XML corpus used in the experiments varies from small to medium sized data and can be accessed from online repository (Moore et al., 2014a). It consists of a variety of data types including string, integer, decimal, numeric-string, IPv4 address, UNIX timestamp and enumeration as shown in table 5.1. The file size ranges from 72 bytes to 5142 bytes. Larger files were not considered as embedded devices on sensor networks do not normally communicate those sizes.

In order to measure energy consumption of data compression, data from two different sources are combined: first the time expended is measured using the GNU/Linux command line tool *time* (GNU, 1999) and second the average energy consumed is measured by using the ODROID smart power device. In order to get reliable and accurate readings, the compression process is repeated 100 times which amortises timing errors across single iterations and also removes one time effects and system level effects. The loop excludes the application initialisation, memory allocation and schema parsing stage in order to accurately calculate only the actual compression time and energy consumption as shown in the algorithm 2 below .

$$time(s) = datasize(bits)/networkspeed(bit/s) \tag{5.2}$$

In order to measure energy consumption of network transfer, it is assumed to be 1.915 Watts per second including the idle consumption which represents energy consumption of typical low-power wireless modules such as 6LoWPAN currently on the market (e.g., Crossbow TelosB) Ayadi et al. (2011a). The data transfer time for communicating the compressed and uncompressed XML files is then calculated using a theoretical network speed using the equation 5.2. Thus it is the fastest/smallest possible time necessary to transfer the given amount of data on the network as factors such as latency, packet loss and fragmentation have not been considered during the calculation. This allows us to obtain accurate energy consumption figures for smaller data sizes. The energy consumption measured for both compression and network transfer activity includes the energy consumed by respective devices when idle.

Algorithm 2 Schema-aware XML compression using (PO, EXI)

$allocate\ memory$
$parse(XMLSchema)$
loop
  $load(XML)$
  $Schema-aware\ compression$
end loop

Algorithm 3 Text-based XML compression using $ZLIB$

$allocate\ memory$
loop
  $load(XML)$
  $Text-based\ compression$
end loop

5.4 Results

In this section, the experimental results are analysed to determine which data compression technique compresses the most and can reduce the total energy consumption after data compression and network transfer. It is assumed that all the files that are compressed on the embedded devices are uncompressed on a server later so the focus is only on the compression and network transfer of the data. The packet size and data rate are assumed to be fixed and thus energy cost of data transfer over network transfer is expected to be linearly

proportional to the size of the data. Thus the energy savings in the data transfer from reduced data sizes must be greater than the extra energy spent on data compression on the device. Hence the compression technique which compresses the most may not always save the most energy (Xu et al., 2003).

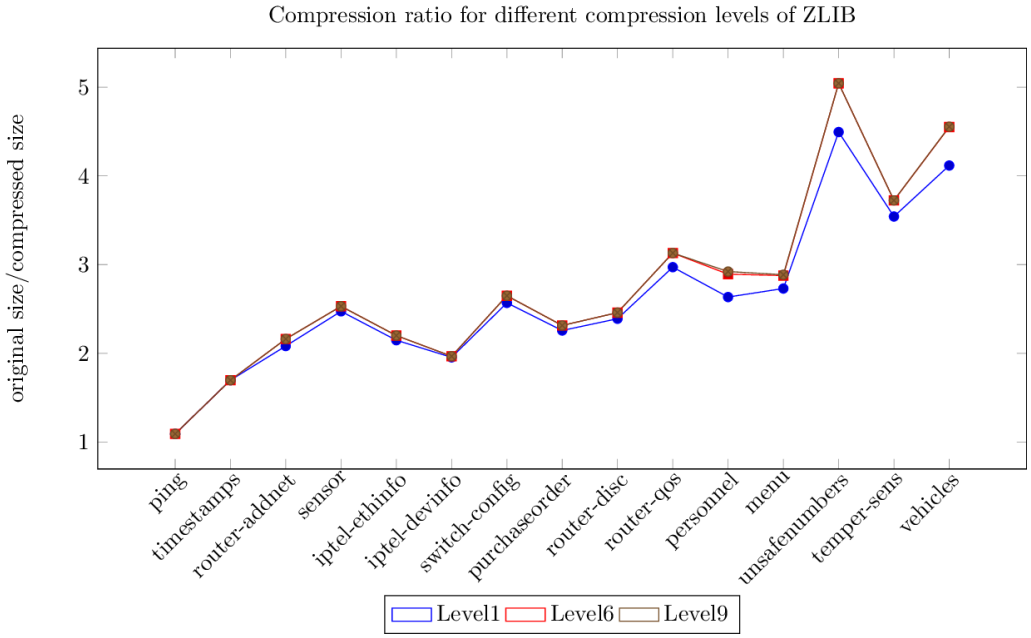

Fig. 5.2 Compression ratio comparison for different compression levels in ZLIB

5.4.1 Compression levels and compression ratio

ZLIB provides 9 different compression levels ranging from 1 (which provides the best speed) to 9 (which provides the best compression ratio). Compression is performed on files discussed above in table 5.1 with 3 levels (1,6 &9) of ZLIB compression. As shown in figure 5.2, the compression ratio for level 9 is only slightly better than level 1 but the compression speed which is shown in figure 5.4 is almost identical for all 3 levels. Hence, the average ZLIB compression level 6 is used during these experiments. Similarly, EXI also provides various options to control the compression ratio, namely default compression, compact compression and schema-aware compression. As shown in figure 5.3, default and compact compression methods consist of similar compression ratios and schema-aware compression has superior compression ratios than the other two techniques. As shown in figure 5.5 schema-aware method spends slightly more time to achieve the extra compression but the vastly su-

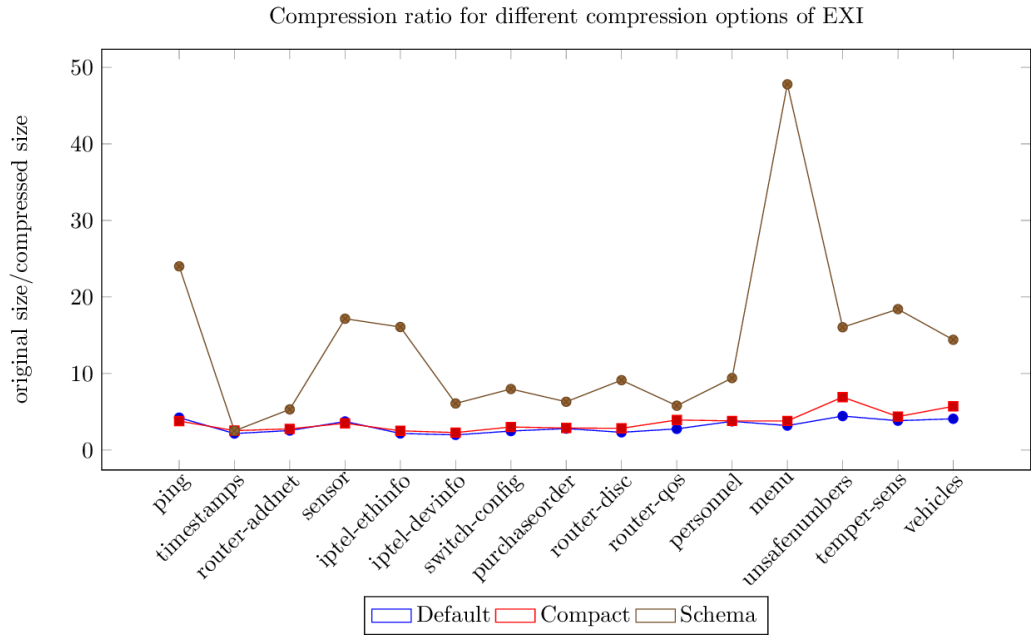

Fig. 5.3 Compression ratio comparison for different compression options in EXI

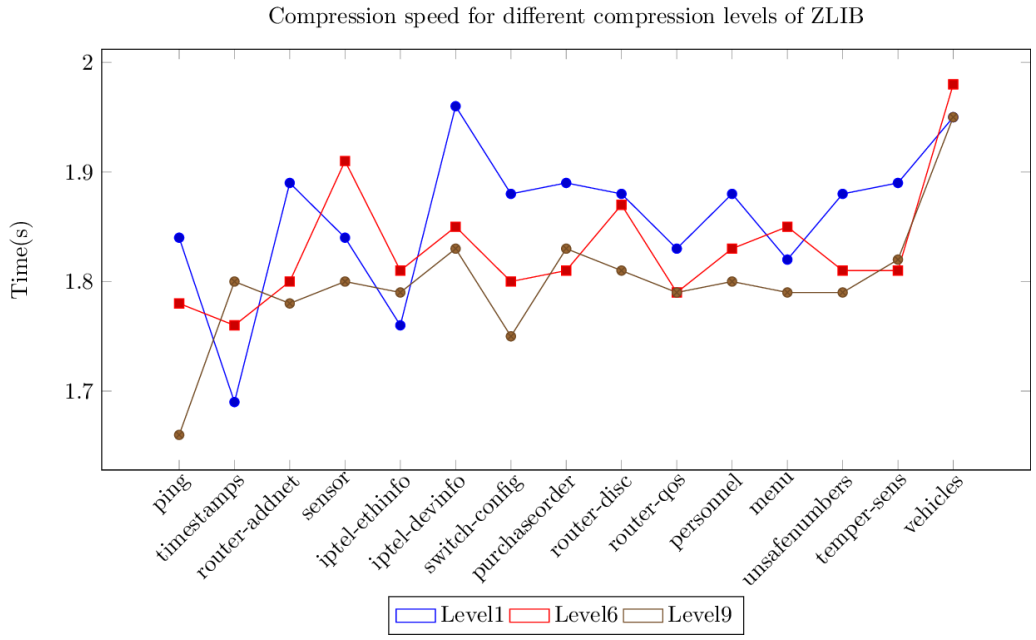

Fig. 5.4 Compression speed comparison for different compression levels in ZLIB

perior compression ratio also justifies its selection. Thus, schema-aware EXI compression is used during the experiments later. PO does not provide different compression levels but data and schema validation can be disabled. However the extra validation requires minimal time compared to the overall

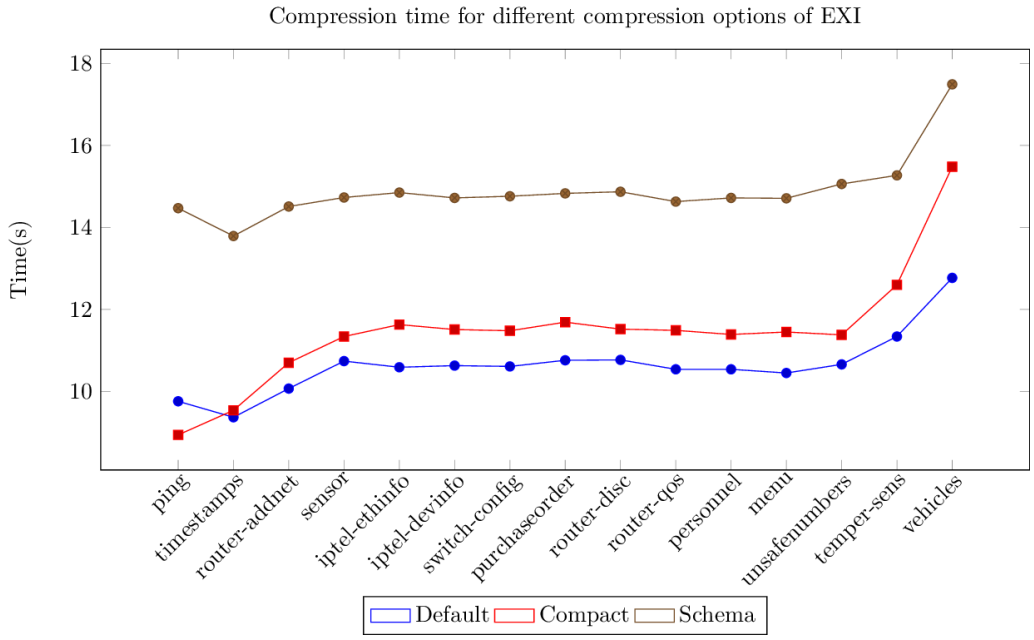

Fig. 5.5 Compression speed comparison for different compression options in EXI

compression process and thus it is enabled during the experiments.

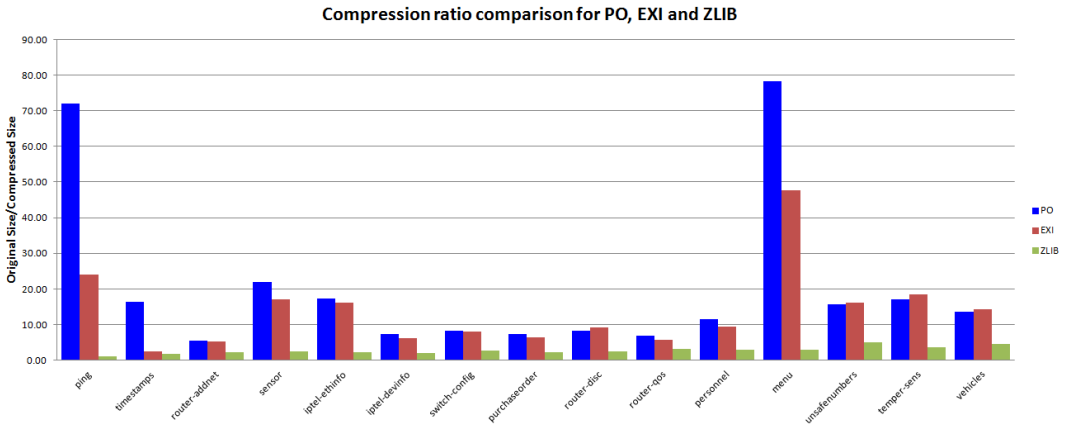

Fig. 5.6 Compression ratio comparison for various compression techniques

In table 5.1 above, the compression ratio for PO, EXI and ZLIB for different XML files are listed in ascending order of size in bytes. It can also be viewed in figure 5.6 where compression ratio is presented as the ratio of uncompressed data size to compressed size and thus higher ratio means better compression and vice versa. PO has the best compression ratio for most of the files and it performs significantly better for files with enumerated and unix-time data types as discussed in Chapter 4.2.2 above. EXI has a com-

pression ratio similar to PO for smaller sized files and slightly better than PO for larger files. ZLIB has the worst compression ratio of the three techniques but it compresses string data types better than non-string types as it uses performs dictionary based compression repetitive strings. The compression ratio does not increase linearly with data size but depends more on the type of the data for PO and EXI. ZLIB's compression ratio does not vary much and is slightly better for larger files due to the occurrence of repeating XML tags.

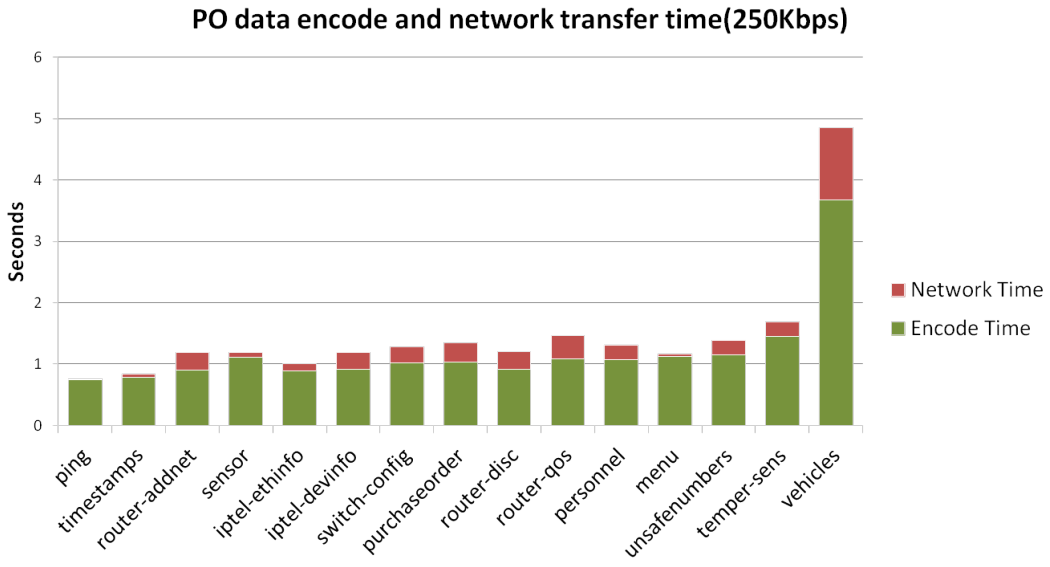

Fig. 5.7 Data encode and network transfer time for PO

5.4.2 Compression speed and data transfer

The data transfer speed used in the experiment is 250kbps to reflect the bandwidth of low-rate IEEE 802.15.4 standard such as 6LoWPAN used by embedded and sensor networks. The transfer time is calculated theoretically as discussed above in methodology section 5.3.2 above. Figures 5.7, 5.8, 5.9 & 5.10 show the compression time for all the files on a Raspberry Pi and transferring the data over a 250kbps network for the three compression techniques along with the uncompressed data transfer. Similarly figure 5.11 shows the combined compression and network trasnfer time for all the four techniques.

The uncompressed data serialisation process requires very small amount of time with compare to network transfer time as shown in figure 5.8 because the serialisation only includes XML to string conversion for network transmission. The network transfer time for the uncompressed data is the biggest

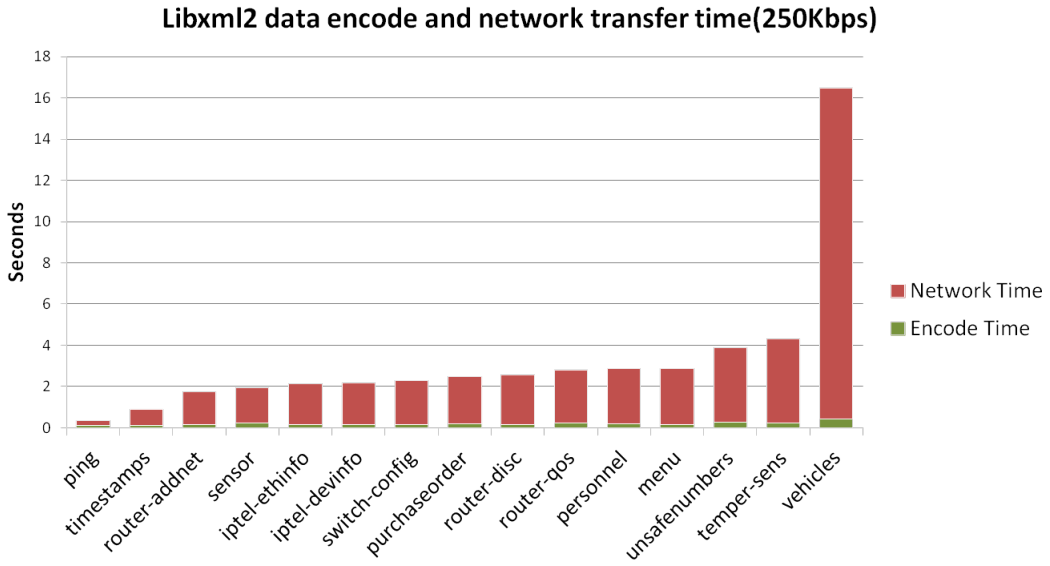

Fig. 5.8 Data encode and network transfer time for LIBXML2

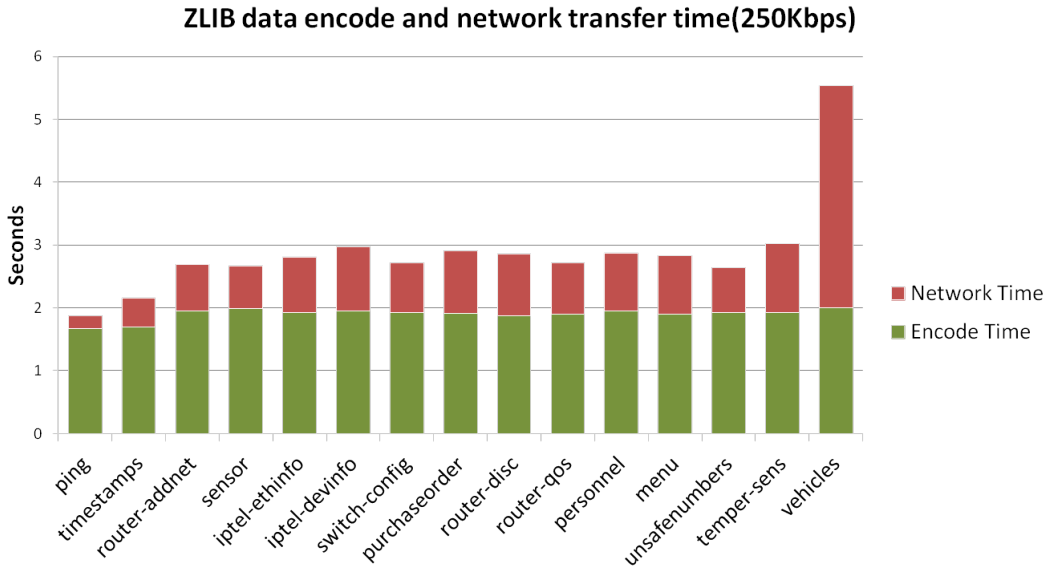

Fig. 5.9 Data encode and network transfer time for ZLIB

compared to the transfer time for any of the three compressed formats. However ZLIB and EXI still take more time on the overall because of the extra time required for compressing the data. Although ZLIB compresses considerably faster than EXI, it still cannot save time overall compared to uncompressed data due to its poor compression ratio. The implementation of EXI used is very ineffective when used on embedded devices despite having an excellent compression ratio due as a result of its extremely slow compression speed. PO

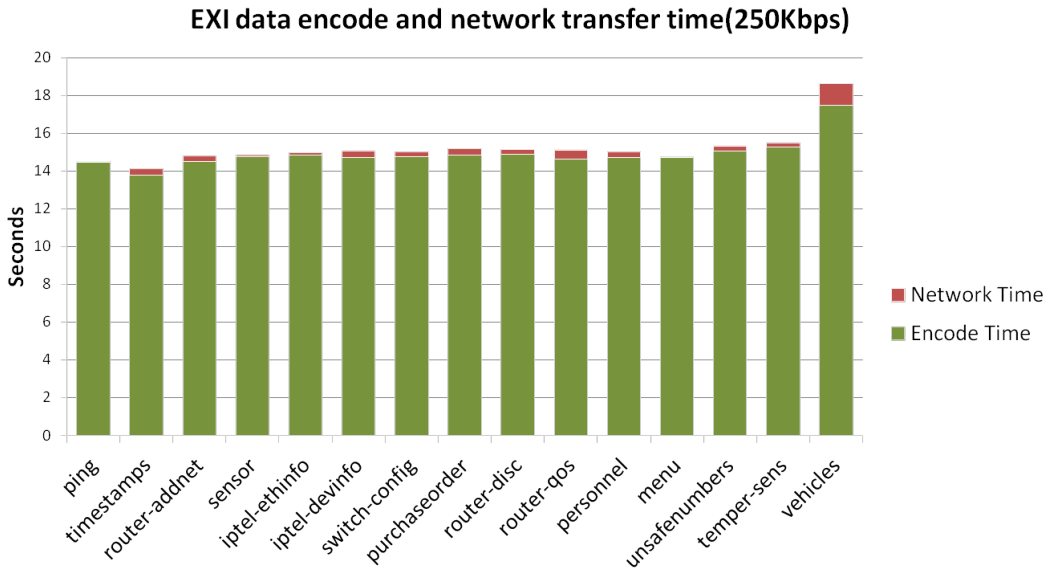

Fig. 5.10 Data encode and network transfer time for EXI

however has the best compression ratio and compresses significantly faster as well. Thus, PO is the best technique which provides time savings overall in terms of compression time and data transfer. The only exception however where PO compression can not provide time savings is for the smallest file which is 72 bytes. This is because of the start up time of PO being bigger than the actual data transfer time of an uncompressed file. Although the time savings for smaller XML files is not huge, it is significant for larger files.

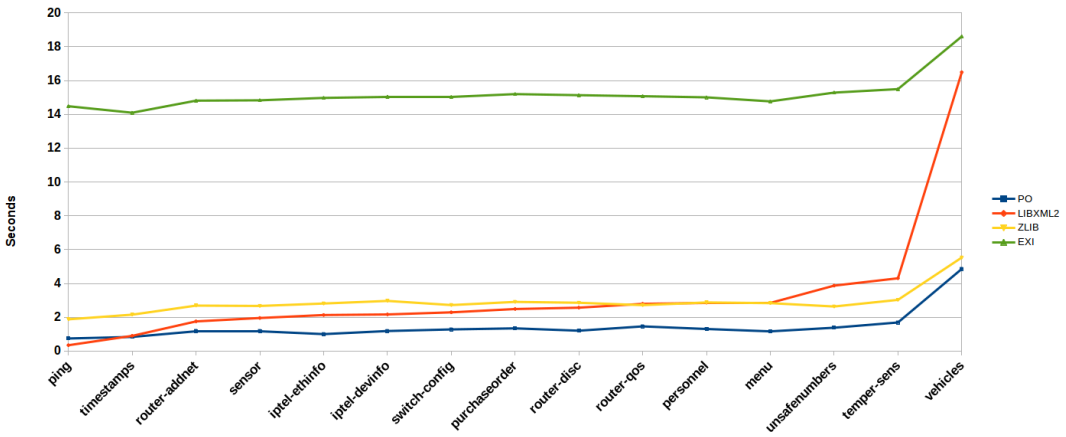

Fig. 5.11 Overall compression and network transfer for all techniques

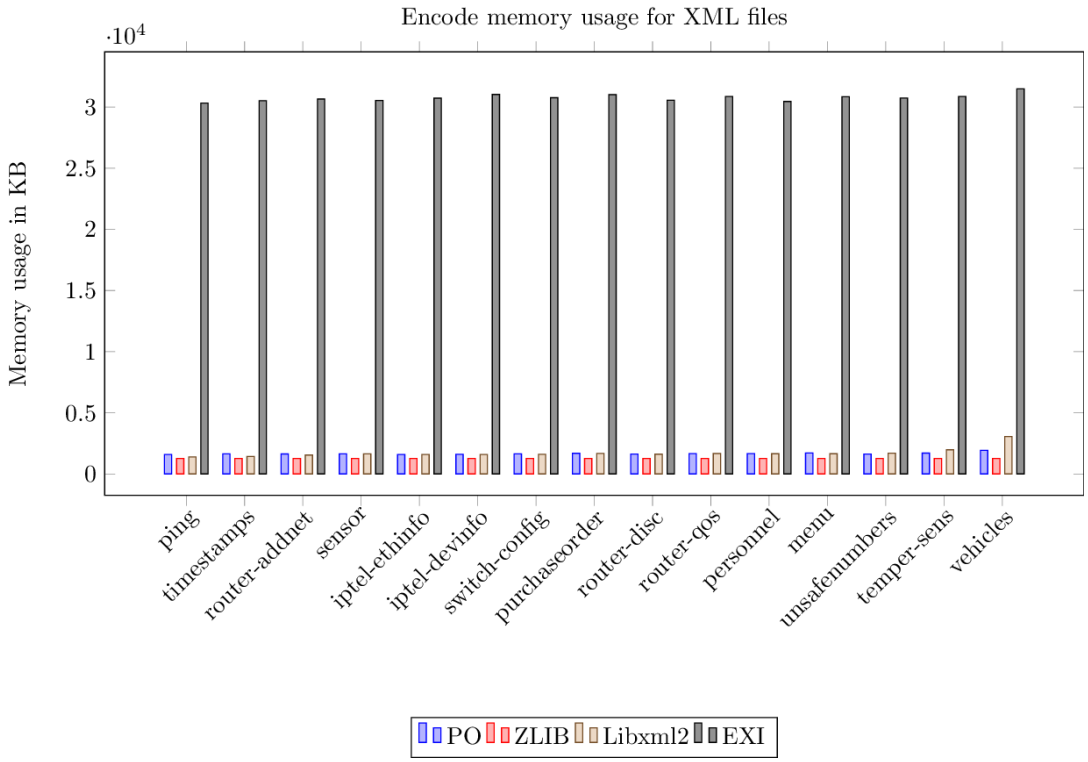

Fig. 5.12 Memory consumption during compression for all techniques

5.4.3 Memory consumption during compression

The memory consumption of the compression techniques is presented in figure 5.12 to determine any impact on compression time and energy consumption. EXI is the most memory hungry tool with consumption around 30000 KB. ZLIB consumes almost constant memory regardless of file size and the consumption stands at 1272 KB. Libxml2 consumes slightly more memory for larger files and its usage ranges from 1392 KB to around 3000 KB. Similarly PO's memory consumption ranges from 1592 KB to 1704 KB and it only varies slightly with file size. The memory consumption depends mostly upon the technique itself rather than the file size used. EXI uses extremely high memory which can be related to its slow compression time too.

5.4.4 Energy cost of compression and data transfer

The average energy consumption per second by the compression techniques in Raspberry Pi is shown in table 5.2. The average energy consumption for network transfer is calculated using typical values for low-power wireless modules. These measurements are used in the calculations as the actual 6LoW-

PAN network has not been implemented yet for use with Raspberry Pi. Both EXI and ZLIB spend more time on the overall than when uncompressed data is communicated over network and energy consumption increases linearly with time. Thus the overall energy consumption of only PO compression and Libxml2 for uncompressed data is shown in figure 5.13 and is calculated using equation 5.1.

Table 5.2 Average energy consumption for 1 second during compression in Joules

Idle	Libxml2	EXI	ZLIB	PO
1.75	1.86	1.96	2.04	2.06

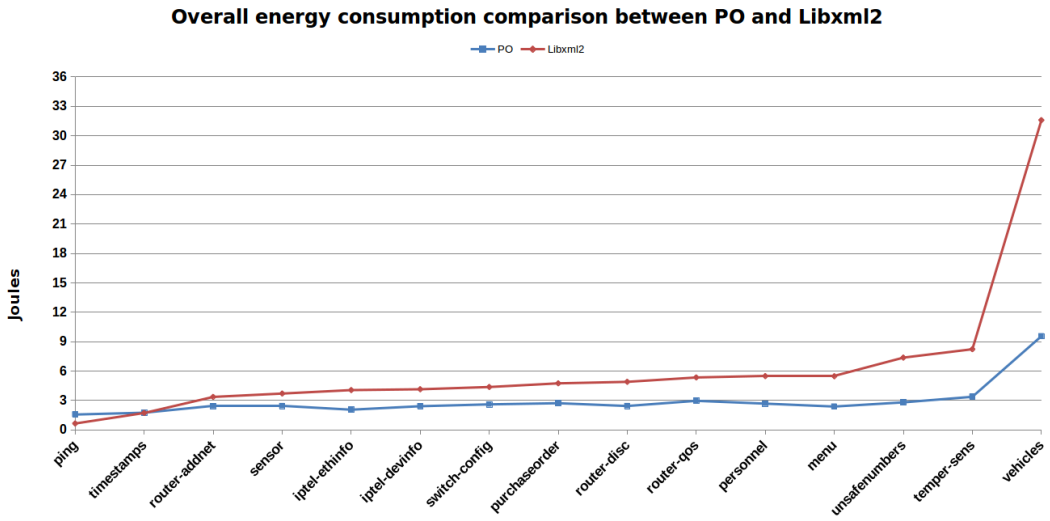

Fig. 5.13 Overall energy comparison for compressed and uncompressed data

PO consumes more energy during compression and less during the data transfer. Libxml2 consumes less energy during data serialisation and more energy during data transfer. However PO consumes less energy on the overall with compare to uncompressed data serialisation and network transfer with the exception of the smallest XML file. The energy savings increases gradually with the increase of file size and the saving is significantly more in the case of the largest file used. PO is saving energy because it consists of the best compression ratio and also the fastest compression speed. EXI and ZLIB fail to save energy because of their slow compression speed.

5.5 Summary

In this chapter the impact of data compression on embedded devices that communicate on a low bandwidth network has been explored and existing schema-aware and text-based data compression for the same domain have been evaluated. Existing research suggested that data compression can only be useful in terms of energy consumption if the energy savings in the data transfer from the reduced data size is greater than the extra energy spent on data compression process. Thus it depends on the available network bandwidth as it can affect the energy cost of data transfer. EXI and ZLIB are not suitable for data compression on embedded devices because they spend too much energy on the compression process which negates the energy savings from the data size reduction in terms of network transfer. PO provides overall energy savings compared to uncompressed data transfer over a 250Kbps network on an embedded device. The energy savings are minimal for files smaller than 800 bytes and significantly bigger for larger files. Thus data compression can save energy when the available network bandwidth is small and it can be useful for large data when the network bandwidth is big. Also data compression techniques which consume too much time for compression such as EXI, and when the compression ratio is not very high such as ZLIB, are not likely to provide energy savings overall.

Chapter 6

Estimating TCP throughput for 6LoWPAN network using a mathematical model

6.1 Background

Embedded devices are becoming more ubiquitous and they are currently used in environment monitoring, asset/logistic tracking and health monitoring (Ahmed et al., 2010; Jafari et al., 2005). The introduction of IPv6 to these tiny devices has enabled them to communicate directly to any other device on the Internet without using any intermediate network/transport layer gateway as a bridge. However, the IEEE 802.15.4 networks physical layer packet size is only 127 bytes and very small payload is available for application data as headers consume significant share of that (IEEE, 2003). As shown in figure below medium access and link layer security use up to 25 and 21 bytes respectively for header leaving only 81 bytes for upper layer. The use of IPv6 at network layer which requires 40 bytes for header leaves only 41 bytes for transport layer. TCP uses minimum of 20 bytes in the header when no TCP options are included leaving only 21 bytes for the application data.

6.2 6LoWPAN and IPV6

Header compression techniques have been already implemented or proposed in order to reduce the header overhead ratio. LOWPAN_IPHC (IPHC) has

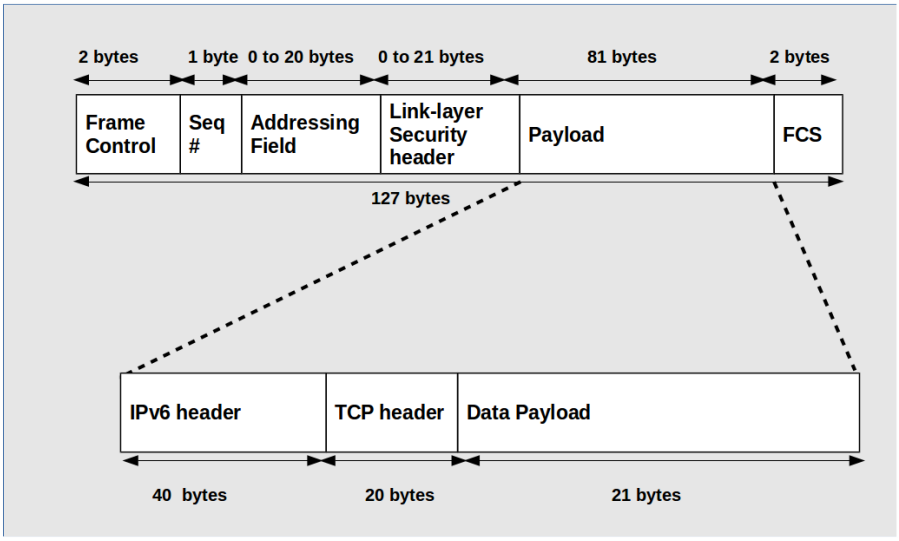

Fig. 6.1 6LoWPAN frame format

been proposed as a new version of the LOWPAN_HC1 IPv6 header compression mechanism which can reduce the IPv6 header to about 3-5 bytes (Hui and Thubert, 2011; Kushalnagar et al., 2007; Montenegro et al., 2007). However such compression can only be achieved when both sender and receiver devices are in the same 6LoWPAN network. IPv6 header can be compressed to about 20 bytes if the sender/receiver is outside the 6LoWPAN network and the external prefix is not known and to about 12 bytes if the external prefix is known (Olsson, 2014). Thus, the IPv6 header can be compressed up to 50% even on the worst case scenario. Similarly UDP (Postel, 1980) header can be compressed from 8 bytes to 4 bytes by omitting fields which can be derived from lower layers (Hui and Thubert, 2011). TCP header compression has not been fully implemented yet but an Internet draft version is being developed by IETF (Ayadi et al., 2010). Ayadi et al. (2011b) discuss the TCP header compression mechanism with more detail which can be used along with LOWPAN_IPHC and can reduce the TCP header size up to 6 bytes which helps to increase the size of application data payload.

Table 6.1 6LoWPAN frame headers without header compression (in bytes)

Total	Mac Layer Header	Link Layer Security	LOWPAN Header	IPv6 Header	TCP Header	Data Payload
127	25	0 (Null)	1	40	20	41
127	25	9 (AES 32bit)	1	40	20	32
127	25	13 (AES 64bit)	1	40	20	28
127	25	21 (AES 128bit)	1	40	20	20

Table 6.2 6LoWPAN frame headers with header compression (in bytes)

Total	Mac Layer Header	Link Layer Security	LOWPAN Header	IPv6 Header	TCP Header	Data Payload
127	25	0 (Null)	1	20	6	75
127	25	9 (AES 32bit)	1	20	6	66
127	25	13 (AES 64bit)	1	20	6	62
127	25	21 (AES 128bit)	1	20	6	54

6.3 6LoWPAN header compression

As shown in table 6.1, if AES 128 bit encryption is used for the link-layer then the application data payload can be as little as 20 bytes with a staggering 84% header overhead. The application data payload is only 41 bytes big even without any link-layer security and the resulting header overhead is still 67% of the 127 bytes frame. Similarly, table 6.2 shows the total header sizes when both IPv6 and TCP header compression can be applied and for different link-layer security options. Now the worst case application data payload increases from 20 bytes to 54 bytes and the best case payload increases up to 76 bytes. The header overhead is only 57% and 40% for the 128 bit and no encryption at link layer respectively. Thus, the application payload data size can range from 20 bytes to 75 bytes depending upon various link-layer security options and header compression techniques.

The actual throughput of the 6LoWPAN network is affected by the application payload size and the packet loss. Although different 6LoWPAN implementations such as Contiki are available, only some of them provide header compression and/or TCP support. It is not yet available on minimal Linux-based OS where data compression can be applied. It is not yet possible to implement compression along with OS which fully supports 6LoWPAN network. Maathis formula as shown in equation 6.1 is most commonly used for TCP throughput estimation but it has some limitations which makes it unsuit-

able for wireless networks. The formula cannot be applied for networks with variable round trip time (RTT) (Mathis et al., 1997) and wireless networks are more likely to have that as a result of high packet loss and dynamic nature of the nodes. Also the formula considers that the packet loss is mainly caused by the TCP congestion avoidance algorithm but typical low bandwidth networks are more likely to implement single TCP window due to their constrained resources. Thus the formula cannot be used to correctly estimate the TCP throughput. So a mathematical model, based on work of Ayadi et al. (2011a), is used to estimate the TCP throughput of a 6LoWPAN network by calculating the expected number of bits to be communicated for a given data frame as a result of packet loss and hop re-transmissions.

$$Throughput = \frac{(C * MSS)}{(RTT * \sqrt{P_{loss}})} \tag{6.1}$$

Where:
C is a constant factor to adjust TCP slow start mechanism
MSS is the maximum segment size
RTT is the round trip time
P_{loss} is the packet loss probability

6.4 TCP throughput estimation on 6LoWPAN networks

Existing researches have studied the TCP performance in multi-hop wireless networks (Ayadi et al., 2011a) and references therein. For example, Galluccio et al. (2003) looked at TCP throughput with an aim to improve such performance measure. Fairhurst and Wood (2002) analysed the impact of the MAC layer protocol on the TCP throughput over wireless multi-hop networks and found that throughput does not increase with window size. Some others have studied various congestion control algorithms over wireless networks (Al Hanbali et al., 2005; Seddik-Ghaleb et al., 2006; Sivakumar and Akyildiz, 2008). Ayadi et al. (2011a) have presented an analytical model to calculate energy consumption during TCP flow in wireless network which consider link-layer and transport-layer acknowledgements and the impact of lower-layer fragmentation, error correction and per-link error rates on the energy-efficiency.

Table 6.3 List of parameters assumed for simplifying of calculation of number of bits sent per TCP segment

Variable	Definition
h	Number of hops between source and destination
r	Maximum number of link-layer transmission attempts
m	Number of fragments corresponding to a single TCP segment (due to link layer fragmentation)
α	FEC redundancy ratio
B	Bit error rate
F_i	Probability that a destination node does not receive a link layer data frame after r attempts (ith hop)
Q_s	Probability of an end-to-end packet transmission success
D	Link-layer data frame size
A	Link-layer acknowledgement frame size
$\mathbf{I_f}$	The expected total number of bits sent for the m fragments to reach the destination, knowing that they (i.e., at least one) finally fail.
$\mathbf{H_f}$	Expected number of bits sent after r attempts knowing that the (one-hop) transmission has failed
$\mathbf{H_s}$	Expected number of bits sent within r attempts knowing that the (one-hop) transmission has succeeded
$\mathbf{E_f}$	Expected number of bits sent for an end-to-end packet transmission knowing that it has failed
$\mathbf{E_s}$	Expected number of bits sent for a successful end-to-end packet transmission knowing that it has succeeded
S	Average number of bits sent for successfully transmitting a TCP segment

The model assumes that the energy consumption depends directly on the number of bits sent by all the nodes in the network. Table 6.3 lists most of the variables used during the calculations (Ayadi et al., 2011a). The equations 6.2 to 6.8 used below are referenced from Ayadi et al. (2011a)'s work.

Some parameters have been assumed for simplifying the calculations and table 6.4 shows the list of such values. The *italics* variable names in the table correspond to probabilities and the **bold** variable names correspond to the expected number of bits. The number of hops (h) in the network is considered to be in the range of 1 to 8 to cover the most commonly used scenarios. The CSMA-CA technique is assumed to be used along with Automatic repeat Request (ARQ) and Forward Error Correction (FEC). The FEC redundancy ratio (α) is assumed to be 0.1 as that is the optimal value for energy consumption and values above result in the increase of the energy consumption due to the

Table 6.4 Default parameters used during the analytical calculations

Parameter	Value
h	1 to 8
r	1
α	$10^{-}1$
B	3.10^{-4}
A	40 bits
IP Header	160 bits
TCP Header	48 bits
D	600 bits

redundancy overhead. The IP header is considered to be of 20 bytes instead of 40 bytes as a result of the header compression (Kushalnagar et al., 2007). Similarly TCP header is considered to be of 6 bytes instead of 20 bytes as a result of the header compression (Ayadi et al., 2011b, 2010).

TCP MSS selection is not a straight forward process. A smaller TCP MSS leads to a larger header to data ratio and can create multiple smaller TCP segments but most TCP segments fit in to a single IP datagram. On the other hand, a large TCP MSS leads to segmentation and can increase the number of end to end re-transmissions but the header to data ratio becomes smaller. Ayadi et al. (2011a) found that short TCP MSS is more suitable for networks with higher packet loss probability and when the maximum number of re-transmission attempt at link-layer (r) is small. Normally long TCP MSS can save more energy when the packet loss probability is small and the link-layer attempt *r* is high. However the packet loss of a typical 6LoWPAN network is at least 5% or greater (Srinivasan et al., 2006) which gives bit error rate of around $3.75.10^{-}4$. Thus a small TCP MSS value of 75 bytes is set so that there is no need for fragmentation at lower layers and the best-case 6LoWPAN MTU is 75 bytes as shown in the table 6.2 above. Also, *r* is assumed to be 1 to provide efficient result along with the selected TCP MSS.

The number of bits sent in a single hop mode in the case of link-layer transmission failure is $H_f := r * D$. The expected number of bits in the case of successful transmission within the *r* attempts can be calculated as below:

$$H_s = \frac{1}{1-F}\left(\sum_{i=1}^{r} P_{partial}^{i} P_{fail}^{r-i}(rD+iA) + \sum_{k=1}^{r} P_{succ} \sum_{i=0}^{k-1} P_{partial}^{i} P_{fail}^{k-1-i}(kD+(i+1)A)\right) \tag{6.2}$$

where,

$$P_{fail} = 1 - \sum_{i=0}^{c} B^i (1-B)^{D-i}$$

$$P_{partial} = (1 - P_{fail})(1 - (1-B)^A)$$

$$P_{succ} = (1 - P_{fail})(1-B)^A$$

The number of bits sent over a multi-hop network for a successful transmission can be calculated with equation 6.3 below by assuming that the transmission on each hop is independent. However if the transmission fails at one of the *h* hops, the expected number of bits depends on the hop where the failure occurred and can be calculated using equation 6.4 below.

$$E_s := 1 - \sum_{i=1}^{h} H_{s_i} \tag{6.3}$$

$$E_f := \frac{\sum_{k=1}^{h} (\sum_{i=1}^{k-1} H_{s_i} + H_{f_k}) \prod_{j=1}^{k=1} (1-F_j) F_k}{1 - \prod_{i=1}^{h} (1-F_i)} \tag{6.4}$$

Now the expected number of bits sent for a TCP data fragment can be calculated using the equations 6.5 and 6.6. For simplicity, each TCP data segment is assumed to be of same size and the TCP re-transmissions is assumed to be unlimited. In order to receive a TCP segment successfully, all the *m* fragments must reach the destination and the TCP acknowledgement must be received. Thus, the expected total number of bits sent by all the nodes is equal to:

$$S_s := E_s * m + E_{s,ack} \tag{6.5}$$

However the expected number of bits sent by all the nodes in the case of failure is:

$$S_f := \frac{1}{1 - Q_s^m * Q_{s,ack}} [I_f (1 - Q_s^m) + (E_s * m + E_{f,ack}) Q_s^m (1 - Q_{s,ack})] \tag{6.6}$$

where,

$$Q_s = \prod_{i=1}^{h}(1 - F_i)$$

$$I_f = m(1 - Q_s)E_f + mE_sQ_s(1 - Q_s^m)$$

6.5 Calculations and results

The TCP window is assumed to be a single TCP segment and it is a reasonable choice for networks with a moderate number of hops (Fairhurst and Wood, 2002). Also, most common TCP implementation on low bandwidth wireless network(e.g. Contiki OS) use a small window and is suitable for the limited processing and memory capabilities of embedded devices (Dunkels et al., 2004). Now, the expected total number of bits sent by all nodes for a successful transmission of a TCP segment and the corresponding acknowledgement can be calculated with equation 6.7.

$$S := S_f(1/P_s - 1) + S_s \tag{6.7}$$

Table 6.5 Expected number of bits sent for a single TCP segment

Number of hops	Expected number of bits sent for MSS of 600 bits	Expected number of bits sent for MSS of 4096 bits
1	1118.12	9660.30
2	2243.52	26460.52
3	4184.08	82930.74
4	7138.24	319766.91
5	11372.89	1351665.62
6	17283.46	5806389.95
7	25355.27	24623046.32
8	36187.31	102423598.85

The total number of bits sent for MSS 75 Bytes (600 bits) and MSS 512 Bytes (4096 bits) is shown in table 6.5. The expected total number of bits to be sent for a user data of given size *M* is the number of TCP segments *[M/MSS]*, multiplied by expected number of bits sent for a successful TCP segment transmission. The results are similar to the original study by Ayadi

et al. (2011a) and short MSSs send smaller amount of data (as a ratio of original data) per TCP segment with compare to long MSSs. The savings from TCP header overhead provided by using long MSSs are exceeded by the cost of end-to-end re-transmissions caused by the higher packet loss.

The effective throughput of the network can now be calculated using the amount of data in a single TCP segment, the amount of bits sent for transmission of a single TCP segment and the theoretical maximum throughput of the network. The estimated effective throughput can be calculated as:

$$Throughput_{effective} := \frac{TCPMSS}{S} * Throughput_{maximum} \tag{6.8}$$

Where, $Throughput_{maximum}$ is the maximum 6LoWPAN (250 Kbps),
TCP MSS is the size of single TCP segment in bits and
S is the total bits sent for transmission of single TCP segment and calculated using equation 6.7

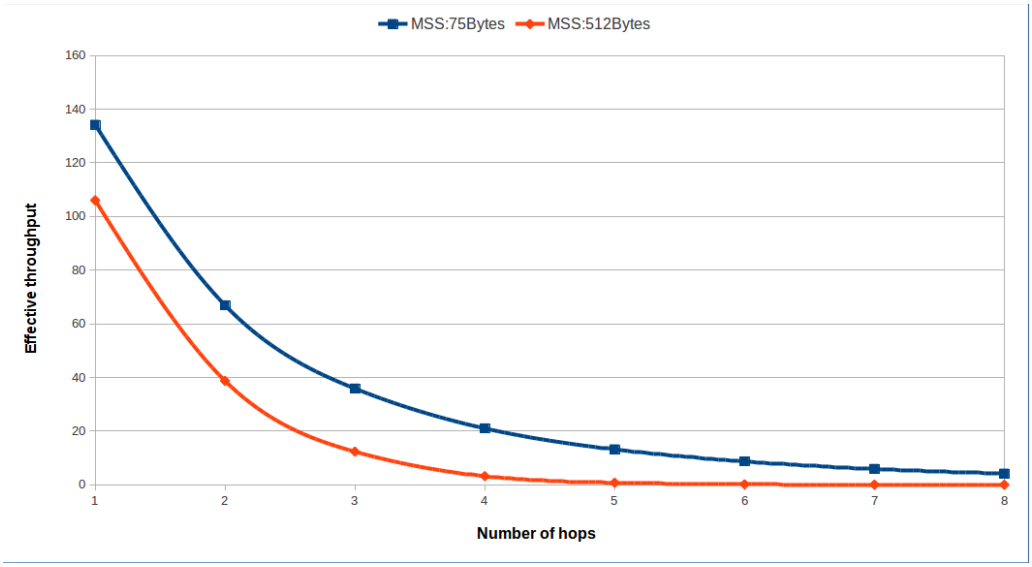

Fig. 6.2 Effective throughput in Kbps as a function of number of hops (BER=3 x$10^{-}4$,$\alpha = 10^{-}1$)

Figure 6.2 shows the effective throughput for two different MSSs as a function of number of hops for a given BER and FEC calculated using equation 6.8 and values from table 6.5. The throughput decreases with the increase in the number of hops and is significantly small for larger number of hops. Short MSSs provide better throughput for a given BER (moderately high) and FEC (small). The advantage is mainly because of the large number of end-

to-end re-transmissions required for large MSSs due to the higher BER which negates the savings provided by a smaller overhead ratio. It must also be noted that the difference between the throughput for short and long MSSs increases with the number of hops. It should not be confused with the actual throughput decreasing in the same scenario. As shown in figure 6.3 short MSS throughput for single hop network is only 1.26 times more than when long MSS is used but it is about 50 times more when there are 6 hops in the network and 400 times for 8 hops network.

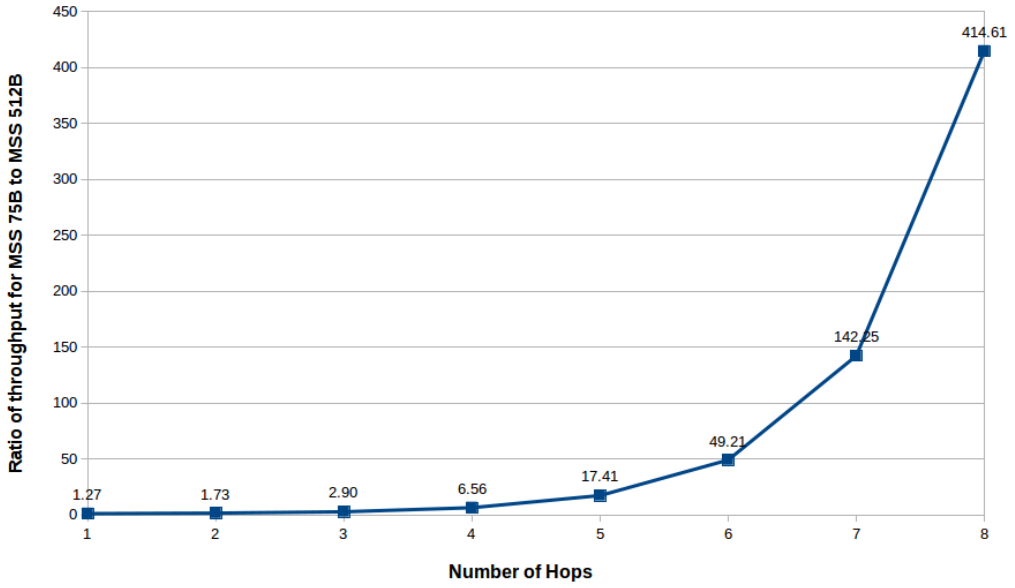

Fig. 6.3 The ratio of throughput for 75B MSS to 512 B MSS (BER=3 x$10^{-}4$,$\alpha = 10^{-}1$)

The estimated throughput for 512 Byte MSS is less than 1Kbps when the number of hops is more than 4. The throughput becomes extremely limited for even small amount of data and thus it is meaningless to create a network with more than 4 hops. However the throughput for 75 Byte MSS with 8 hops is still 4 Kbps and is more than the throughput provided by 512 Byte MSS with just 4 hops.

In order to analyse the impact of the reduced throughput on the overall time spent by various compression techniques discussed in Chapter 5, the average of the XML file sizes and compression time for various techniques is calculated. The average is used so that the throughput estimation for different number of hops can be used in a simplified manner. As shown in figure 6.5, the bandwidth reduction because of the number of hops is significantly af-

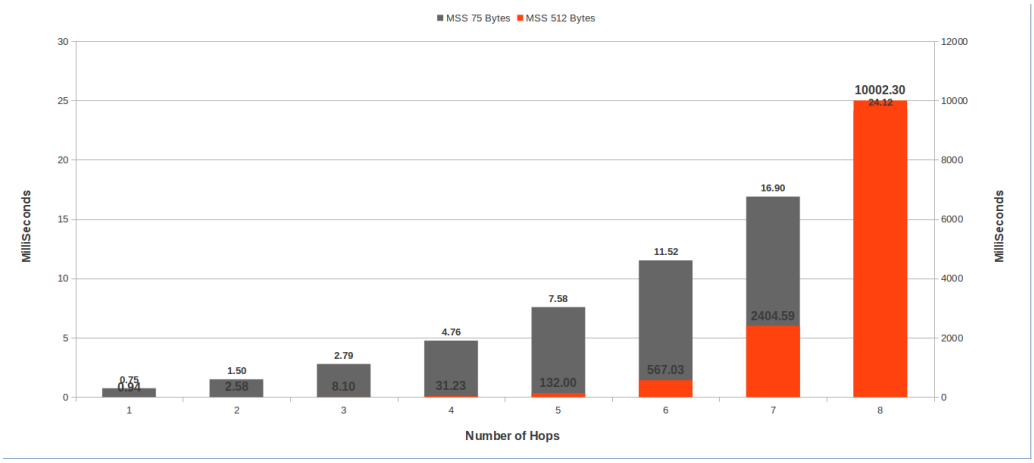

Fig. 6.4 Time taken to transfer 100 bits data with 75B and 512B MSS (BER=3 x$10^{-}4, \alpha = 10^{-}1$)

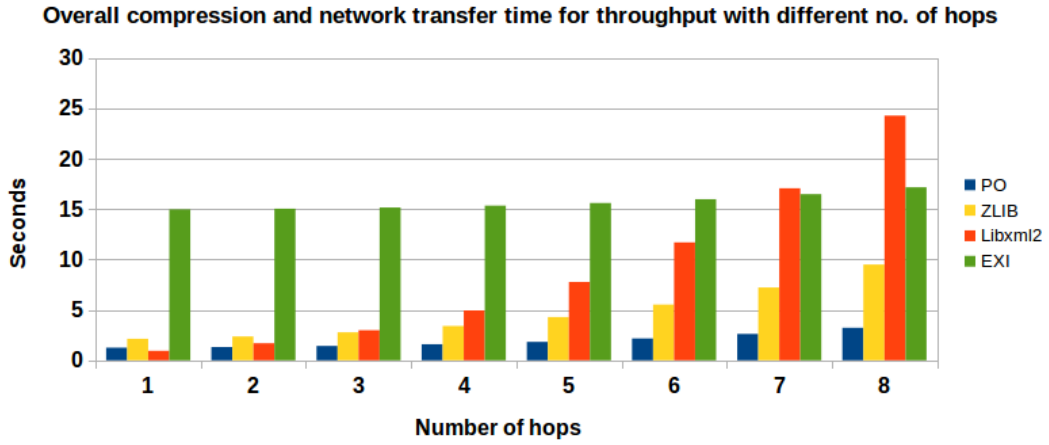

Fig. 6.5 Overall time comparison for compression and network transfer for average XML files

fecting the compression techniques with poor compression ratio and uncompressed data transfer. The compression time remains constant regardless of the actual bandwidth so the overall time taken depends heavily up on the actual network transfer time. Thus the overall time taken increases with the number of hops for Libxml2 and ZLIB. On the other hand the overall time taken remains almost same for PO and EXI. The actual data size which is communicated over the network is very small in the case of these two techniques due to their superior compression ratio and thus the decrease in throughput is not affecting the overall time like for Libxml2 and ZLIB. EXI spends the most time on the overall except when there are 7 or more hops and it's due to the compression time being bigger than the actual network transfer time of the

uncompressed data. PO consists of the smallest overall time except when the network contains just a single hop. Otherwise it is always beneficial to compress data and the time savings increase with the increase in the number of hops or with the reduction in throughput.

Finally, the overall energy consumption from data compression and network transfer for average XML data set is shown in figure 6.6. The values used for the calculations are presented in table 6.6. The overall energy consumption of uncompressed data is slightly smaller than the compressed data in case of a single hop network. The energy consumption of uncompressed data increases significantly with the increase in the number of hops while the energy consumption of compressed data increases very slowly. As a result the energy savings from compression increases with the number of hops and thus it can be concluded that compression is beneficial for networks with more than 1 hop and becomes more useful for larger networks.

Table 6.6 Average energy consumption for 1 second during compression and network transfer in Joules

Idle	Libxml2	PO	Network
1.75	1.86	2.04	1.915

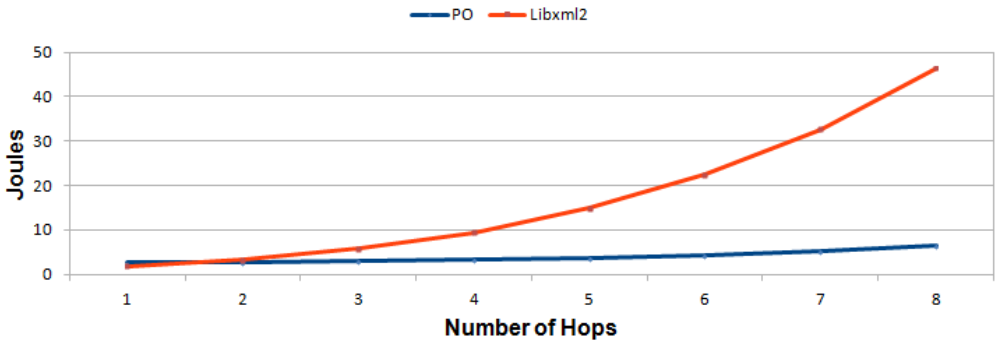

Fig. 6.6 Overall energy consumption comparison between PO and Libxml2 for various number of hops

6.6 Summary

In this chapter a mathematical model is used to estimate the TCP throughput of a 6LoWPAN network by considering the MAC layer MTU, TCP MSS, bit error rate, FEC redundancy ratio and re-transmissions and number of hops in the network. The estimation was performed by calculating the possible number of bits which is sent over the network to successfully transmit a TCP segment. The calculation was carried out for TCP MSS of 75 Bytes which can be fit into a single 6LoWPAN MTU without segmentation and TCP MSS of 512 Bytes in order to highlight the suitability of smaller MSS. The effective TCP throughput is calculated by using the total number of bits transmitted for a single TCP segment, the TCP segment size and the maximum TCP throughput of the network. A smaller TCP MSS is found to be more suitable for 6LoWPAN network especially as the network size grows with more hops.

The estimated TCP throughput is used to improve the results discussed in Chapter 5. The overall time taken to communicate data compressed with different compression techniques and uncompressed data is recalculated by using the newly estimated throughput. It is found that the overall time for compressed data is not affected much by the increase in the size of the network but the time for uncompressed data increases significantly. Similarly the energy consumption analysis showed that it is not beneficial to use compression on a single hop network and it is increasingly important to compress data as the network size grows.

Chapter 7

Conclusions

This research investigated the impact of data compression on the overall performance of low-rate low-power wireless mesh networks. The potential of such networks to regularly communicate data in real-time and the associated challenges were highlighted. The research focused on XML to be used as highly structured data format mainly because of its popularity in terms of existing usage, portability to embedded hardware and robust libraries for processing and meta-data representation. However this research can be extended to any other highly structured data format suitable for embedded devices. Similarly ZeroMQ is used as the messaging protocol to provide simple publish/subscribe messaging model mainly because of the portability of software and light-weight architecture. The messaging protocol can be easily replaced with alternatives that can improve the current results. Schema-aware and text-based compression techniques are evaluated in this research and light-weight techniques, which can provide efficient compression ratio without spending too much time, have been considered the best.

7.1 Summary of the Thesis

In this thesis the potentials and the challenges of using wireless mesh networks to create a real-time yet scalable data sharing system are explored. The research is motivated by the increasing popularity of wireless sensor networks to connect the physical world directly to the Internet. The sensors are becoming smarter and are capable of sensing, processing and communicating intelligent information which can be used to assist humans in a variety of scenarios. A collection of smaller sensor nodes can be used to improve

or replace existing large standalone sensor systems which are often expensive. A prototype wireless testbed was successful in collecting, processing and communicating accelerometer sensor data (to detect seismic activities) but the limited network bandwidth and frequent data communication severely affected the scalability of the prototype. Hence the research focused on improving the network performance by reducing the data sent over the network and evaluated existing techniques which are suitable for such networks.

Firstly, light-weight, portable and scalable message communication models and messaging protocols are studied. The ideal protocol should be capable of running on embedded hardware with constrained resources and also provide messaging without adding extra control overhead. ZeroMQ is found to be the most suitable protocol and is used along with Pub/Sub model. Then a data sharing system is created by combining highly structured data format, messaging protocol and data compression technique. XML is used to represent the sensor data as they are structured and predictable by nature. XML schema is used to define the data structure and restrictions of data type, range and choices. The use of schema is important aspect of the data format as it is used to efficiently compress the data. Existing text-based and schema-based data compression techniques are evaluated to determine the most efficient way of compressing the data on an embedded device. Data compression has been found to be beneficial as it reduces data sent over network. However the compression process must not consume other resources, mainly time and energy, in order to save bandwidth usage.

The trade-off between data compression speed, data size reduction and energy consumption is a complex affair as it depends on the embedded hardware's resources, the compression technique's efficiency and the available network bandwidth of the wireless network. The study found that it is beneficial to compress data if the compression ratio is efficient and compression speed is fast as well. The compression ratio alone should not be used to judge a compression technique as the compression speed can make it slower on the overall than uncompressed data. The EXI implementation provided efficient data compression ratio however the compression speed was slow making it unsuitable for these kind of devices. ZLIB, which is one of the best text-based compression techniques, boosts a fast compression speed but fails to provide the compression ratio efficiency like schema-aware techniques and thus fails to be useful. PO provides efficient compression with reasonably fast speed and saves time with compared to communicating uncompressed data. Simi-

larly the energy consumption of PO compression is compared against the energy consumption of uncompressed data transfer and compression is found to be beneficial except for a small-sized data. The overall energy consumption is studied in the case of wireless networks of different size (number of hops). The TCP throughput of such networks is estimated by calculating the total bits of data sent while transmitting a single TCP segment. The results found that the PO compression is beneficial to save the overall energy in wireless networks consisting of more than single hop. The savings increase significantly as the number of hops increases as the actual network bandwidth decreases.

7.2 Technical contributions

The main technical contributions of this thesis are presented below.

Messaging models

This research studied different messaging protocols to determine a suitable messaging model to communicate sensor data over wireless network. Sensor network consist of dynamic nodes and thus the messaging model should support the nodes joining/leaving regularly. Request/Reply, Remote procedure call and Publish/Subscribe messaging models were studied in the context of sensor networks. Publish/Subscribe model can be extended to become fully independent of sender/receiver's existence by adding intermediate broker. A sensor node can send data to the broker before any receiver node is connected. The receiver node can get the data from broker even after the sender is disconnected. The network congestion for broker-based and broker-less models were analysed. Broker-based model solves the dynamic discovery issue of sensor nodes and is simpler to manage than broker-less model. It can be reused to create sensor data networks in a scalable way.

Efficient sensor data compression

Sensor data can be represented in a range of formats such as text, image, sound or video. Data compression in sensor networks is not new but text based data can be represented in highly structured format in order to compress it efficiently. This type of compression is relatively new in sensor net-

works and can be used to reduce bandwidth usage without spending constrained resources. This research studied different XML compression techniques to determine the best way of reducing data size. The compression techniques were selected mainly based on their ability to run on embedded devices with limited resources. This included the study of Packedobjects and EXI as XML schema-aware compression techniques. ZLIB was included as the most efficient text-based compression technique. And Libxml2 was used as technique to pass uncompressed data to the network. The compression speed (time) and compression ratio (original size/compressed size) were used as benchmark to compare them.

The best compression technique is generally expected to be the one which compresses the data most. However the compression speed is equally important as the compressed data also needs to be communicated over a network. Thus the overall compression and network transfer time of compressed data must be smaller than the network transfer time of uncompressed data. The network bandwidth is also important aspect of the evaluation as it determines the actual data transfer time. Thus, technique with the best compression ratio may not be the most suitable one for embedded devices. The results provide the analysis of efficiency of different XML compression techniques in the context of embedded devices and wireless networks. The results allowed to understand the situations where compression can be successfully applied in order to save bandwidth usage and the total data transfer time when communicating sensor data. The findings of this contribution can be used as a reference for other researchers and developers working in the area of sensor data sharing.

Energy consumption trade-off for XML compression

This research studied the energy cost of XML compression techniques to illustrate any trade-offs for the benefits of network bandwidth savings. The energy consumption of compression process is measured using a physical hardware. The energy consumption of data transfer is calculated from the transfer time and typical energy cost of the wireless standard. The results demonstrate that the data compression on embedded devices is not always energy efficient. Compression speed is one of the main factors which can determine if the energy cost overtakes that of uncompressed data transfer. Compression ratio is also equally important as it affects the transfer time of

the compressed data. In terms of compression techniques, the compression speed of techniques targeted for embedded devices is better than general purpose technique. The results can be used to determine benefits of structured data compression on sensor networks.

7.3 Limitations

This research focused on impact of data compression on the overall performance of wireless mesh network by looking at the network bandwidth usage, data transfer time, compression time and respective energy consumption. Schema-aware XML compression was considered but it can be thoroughly analysed to study the possibility to adjust the compression time to improve performance especially in the case of files smaller than 100 bytes. This allows the reduction of data compression time in case of such files.

This research performed energy consumption measurement for compression techniques using external power consumption monitoring device. The measurement is mainly analysed in relation to the time elapsed however further analysis could be done to study impact of memory and CPU usage. The energy consumption of data transfer over network is calculated theoretically due to the lack of low bandwidth network implementation on the selected device/operating system. The accuracy could be improved by performing the measurement with real network device. Similarly the bandwidth estimation of 6LoWPAN network can also be improved by using actual implementation on the selected device as it is not currently implemented on general purpose operating systems such as embedded Linux.

7.4 Future Work

Future work will focus on implementing the data sharing system to create wireless sensor networks for specific use cases. The implementation can be used to analyse the improvements of the overall performance of such networks when using schema-aware data compression. It will also provide platform to compare the results of the mathematical model in terms of the actual network bandwidth achieved. Further work can be done to improve various components of the data sharing system. Newer message middle-ware can be studied to analyse if they can provide efficient networking than the current

implementation, ZeroMQ. For example, MQTT has been increasingly used in the context of wireless sensor networks and can be considered. Similarly, message formats such as JSON can be studied along with their schema to evaluate the possibility to adding them to supported data formats or to even replace XML if the overall performance can be improved significantly.

References

Aghdasi, H. S., Abbaspour, M., Moghadam, M. E., and Samei, Y. (2008). An energy-efficient and high-quality video transmission architecture in wireless video-based sensor networks. *Sensors*, 8(8):4529–4559.

Ahmad, J., Khan, H., and Khayam, S. (2009). Energy efficient video compression for wireless sensor networks. In *Information Sciences and Systems, 2009. CISS 2009. 43rd Annual Conference on*, page 629–634.

Ahmed, N., Rutten, M., Bessell, T., Kanhere, S., Gordon, N., and Jha, S. (2010). Detection and Tracking Using Particle-Filter-Based Wireless Sensor Networks. *Mobile Computing, IEEE Transactions on*, 9(9):1332–1345.

Akyildiz, I. F., Melodia, T., and Chowdhury, K. R. (2007). A Survey on Wireless Multimedia Sensor Networks. *Comput. Netw.*, 51(4):921–960.

Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., and Cayirci, E. (2002). Wireless Sensor Networks: A Survey. *Comput. Netw.*, 38(4):393–422.

Al Hanbali, A., Altman, E., and Nain, P. (2005). A survey of TCP over ad hoc networks. *IEEE Communications Surveys and Tutorials*, 7(1-4):22–36.

Allen, R. (2008). At First Jolt: Will we have warnings for the next big earthquake. *Geotimes*, 53(10):52–59.

Allen, R. M. and Kanamori, H. (2003). The potential for earthquake early warning in southern California. *Science*, 300(5620):786–789.

Almesberger, W. (2012). Ben WPAN. http://en.qi-hardware.com/wiki/Ben_WPAN.

Amir, Y., Danilov, C., Miskin-Amir, M., Schultz, J., and Stanton, J. (2004). The Spread Toolkit: Architecture and Performance. http://www.cnds.jhu.edu/pub/papers/cnds-2004-1.pdf. Last accessed: 08/11/2009.

Ayadi, A., Maille, P., and Ros, D. (2011a). TCP over Low-Power and Lossy Networks: Tuning the Segment Size to Minimize Energy Consumption. In *New Technologies, Mobility and Security (NTMS), 2011 4th IFIP International Conference on*, page 1–5.

Ayadi, A., Maille, P., Ros, D., Toutain, L., and Zheng, T. (2011b). Implementation and evaluation of a TCP header compression for 6LoWPAN. In *Wireless Communications and Mobile Computing Conference (IWCMC), 2011 7th International*, page 1359–1364.

Ayadi, A., Ros, D., and L., T. (2010). TCP header compression for 6LoWPAN. draft-aayadi-6lowpan-tcphc-01.

Bagale, J. (2010). Fake 2D accelerometer data. https://gitorious.org/fakeshake.

Bagale, J., Moore, J., Kheirkhahzadeh, A., and Komisarczuk, P. (2012). Comparison of Messaging Protocols for Emerging Wireless Networks. In *New Technologies, Mobility and Security (NTMS), 2012 5th International Conference on*, pages 1–5.

Barr, K. C. and Asanovic, K. (2006). Energy-aware Lossless Data Compression. *ACM Trans. Comput. Syst.*, 24(3):250–291.

Birrell, A. D. and Nelson, B. J. (1984). Implementing remote procedure calls. *ACM Transactions on Computer Systems (TOCS)*, 2(1):39–59.

Case, J., Fedor, M., Schoffstall, M., and Davin, J. (1988). Simple Network Management Protocol. RFC 1067. Obsoleted by RFC 1098.

Chang, F., Dean, J., Ghemawat, S., Hsieh, W. C., Wallach, D. A., Burrows, M., Chandra, T., Fikes, A., and Gruber, R. E. (2008). Bigtable: A Distributed Storage System for Structured Data. *ACM Trans. Comput. Syst.*, 26(2):4:1–4:26.

Chen, M., Gonzalez, S., Vasilakos, A., Cao, H., and Leung, V. (2011). Body area networks: a survey. *Mobile Networks and Applications*, 16(2):171–193.

Clayton, R. W., Heaton, T., Chandy, M., Krause, A., Kohler, M., Bunn, J., Guy, R., Olson, M., Faulkner, M., Cheng, M., et al. (2012). Community seismic network. *Annals of Geophysics*, 54(6).

Cochran, E., Lawrence, J., Christensen, C., and Jakka, R. (2009). The quake-catcher network: Citizen science expanding seismic horizons. *Seismological Research Letters*, 80(1):26.

Collins, T. and Moore, J. (2010). An inverse sensor model for earthquake detection using mobile devices. International Conference on Informatics in Control, Automation and Robotics (ICINCO 2010).

Culler, D. (2008a). BLIP. http://smote.cs.berkeley.edu:8000/tracenv/wiki/blip.

Culler, D. (2008b). Contiki uIPv6. http://contiki.sourceforge.net/docs/2.6/a01789.html.

Deering, S. and Hinden, R. (1998). Internet Protocol, Version 6 (IPv6) Specification. RFC 2460 (Draft Standard). Updated by RFCs 5095, 5722, 5871, 6437, 6564, 6935, 6946, 7045, 7112.

Dunkels, A., Gronvall, B., and Voigt, T. (2004). Contiki-a lightweight and flexible operating system for tiny networked sensors. In *Local Computer Networks, 2004. 29th Annual IEEE International Conference on*, page 455–462. IEEE.

Dworak, A., Charrue, P., Ehm, F., Sliwinski, W., and Sobczak, M. (2011). Middleware Trends And Market Leaders 2011. *Conf. Proc.*, C111010(CERN-ATS-2011-196):FRBHMULT05. 4 p.

Estrin, D., Govindan, R., Heidemann, J., and Kumar, S. (1999). Next century challenges: Scalable coordination in sensor networks. In *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, page 263–270. ACM.

Eugster, P., Felber, P., Guerraoui, R., and Kermarrec, A. (2003). The many faces of publish/subscribe. *ACM Computing Surveys (CSUR)*, 35(2):114–131.

Fairhurst, G. and Wood, L. (2002). Advice to link designers on link Automatic Repeat reQuest (ARQ). RFC 3366 (Best Current Practice).

Gailly, J.-L. and Adler, M. (1995). zlib. http://zlib.net/.

Galluccio, L., Morabito, G., and Palazzo, S. (2003). An analytical study of a tradeoff between transmission power and FEC for TCP optimization in wireless networks. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, volume 3, page 1765–1773. IEEE.

Garrett, C. (22 Mar 2012). EXIProcessor. http://sourceforge.net/p/exiprocessor/home/Home/.

Gil, B. and Trezentos, P. (2011). Impacts of Data Interchange Formats on Energy Consumption and Performance in Smartphones. In *Proceedings of the 2011 Workshop on Open Source and Design of Communication*, OSDOC '11, page 1–6, New York, NY, USA. ACM.

GNU (1999). CPU time enquiry. http://www.gnu.org/software/libc/manual/html_node/CPU_Time.html.

Halevy, A. (2010). Structured Data on the Web. In *Web Conference (APWEB), 2010 12th International Asia-Pacific*, page 7–7.

Hao, Y. and Foster, R. (2008). Wireless body sensor networks for health-monitoring applications. *Physiological Measurement*, 29(11):R27.

Hintjens, P. (2007). ZeroMQ. http://www.zeromq.org.

Hintjens, P. (2010). Network traffic monitoring with ZeroMQ. http://zeromq.org/whitepapers:traffic-monitoring-v20.

Hintjens, P. (2011). ZeroMq - The guide. http://zguide.zeromq.org/.

Hintjens, P. (2012). Who is using ZeroMQ? http://zeromq.org/intro:read-the-manual.

Huang, Y. and Garcia-Molina, H. (2004). Publish/Subscribe in a Mobile Environment. *Wirel. Netw.*, 10(6):643–652.

Hui, J. and Thubert, P. (2011). Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks. RFC 6282 (Proposed Standard).

Hunkeler, U., Truong, H., and Stanford-Clark, A. (2008). MQTT-S—A publish/subscribe protocol for Wireless Sensor Networks. In *Communication Systems Software and Middleware and Workshops, 2008. COMSWARE 2008. 3rd International Conference on*, page 791–798. IEEE.

IEEE (2003). IEEE 802.15 WPAN task group 4. http://www.ieee802.org/15/pub/TG4.html.

ISO8879:1986 (1986). Information processing – Text and Office Systems – Standard Generalized Markup Language (SGML). Standard No. ISO 8879:1986, International Organization for Standardization.

Jacobson, V., Leres, C., and McCanne, S. (1989). The tcpdump manual page. *Lawrence Berkeley Laboratory, Berkeley, CA*.

Jafari, R., Encarnacao, A., Zahoory, A., Dabiri, F., Noshadi, H., and Sarrafzadeh, M. (2005). Wireless sensor networks for health monitoring. In *Mobile and Ubiquitous Systems: Networking and Services, 2005. MobiQuitous 2005. The Second Annual International Conference on*, page 479–481.

Kelley, R., Gupta, A., Kumar, A., and Heragu, S. (2011). Using UICDS to Share Data in the Real-Time Decision Support System for Pandemic Response. In *Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 2011 International Conference on*, page 601–605.

Kheirkhahzadeh, A., Moore, J., and Bagale, J. (2013). XML-compression techniques for efficient network management. In *Globecom Workshops (GC Wkshps), 2013 IEEE*, pages 996–1000.

Kim, S., Pakzad, S., Culler, D. E., Demmel, J., Fenves, G., Glaser, S., and Turon, M. (2007). Health monitoring of civil infrastructures using wireless sensor networks. In Abdelzaher, T. F., Guibas, L. J., and Welsh, M., editors, *IPSN*, page 254–263. ACM.

Krishnamurthy, L., Adler, R., Buonadonna, P., Chhabra, J., Flanigan, M., Kushalnagar, N., Nachman, L., and Yarvis, M. D. (2005). Design and deployment of industrial sensor networks: experiences from a semiconductor plant and the north sea. In Redi, J., Balakrishnan, H., and Zhao, F., editors, *SenSys*, page 64–75. ACM.

Kulkarni, A. M., Thirunavukkarasu, J., Pillai, P. S., Sulegai, S. S., and Rao, S. (2012). Insertion and Querying Mechanism for a Distributed XML Database System. In *Proceedings of the 5th ACM COMPUTE Conference: Intelligent & Scalable System Technologies*, COMPUTE '12, page 10:1–10:8, New York, NY, USA. ACM.

Kushalnagar, N., Montenegro, G., and Schumacher, C. (2007). IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals. RFC 4919 (Informational).

Lai, X., Liu, Q., Wei, X., Wang, W., Zhou, G., and Han, G. (2013). A survey of body sensor networks. *Sensors*, 13(5):5406–5447.

Latre, B., Braem, B., Moerman, I., Blondia, C., and Demeester, P. (2011). A survey on wireless body area networks. *Wireless Networks*, 17(1):1–18.

Lembo, S., Kuusisto, J., and Manner, J. (2010). In-depth breakdown of a 6LOWPAN stack for sensor networks. *International Journal of Computer Networks & Communications (IJCNC)*, 2(6).

Levis, P., Madden, S., Polastre, J., Szewczyk, R., Whitehouse, K., Woo, A., Gay, D., Hill, J., Welsh, M., Brewer, E., et al. (2005). TinyOS: An operating system for sensor networks. In *Ambient intelligence*, page 115–148. Springer.

Ley, K., Comer, K., Counihan, J., Czarnecki, D., Foster, S., Husain, H., and McCrabb, R. (2007). Developing a Test-bed for Distributed Search by Mobile Sensors. In *Systems and Information Engineering Design Symposium, 2007. SIEDS 2007. IEEE*, page 1–6.

Li, K. and Hudak, P. (1989). Memory coherence in shared virtual memory systems. *ACM Transactions on Computer Systems (TOCS)*, 7(4):321–359.

Lindemann, C. and Waldhorst, O. (2002). A distributed search service for peer-to-peer file sharing in mobile applications. In *Peer-to-Peer Computing, 2002. (P2P 2002). Proceedings. Second International Conference on*, page 73–80.

Mammeri, A., Hadjou, B., and Khoumsi, A. (2012). A survey of image compression algorithms for visual sensor networks. *ISRN Sensor Networks*, 2012.

Mathis, M., Semke, J., Mahdavi, J., and Ott, T. (1997). The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm. *SIGCOMM Comput. Commun. Rev.*, 27(3):67–82.

Mazzer, Y. and Tourancheau, B. (2009). Comparisons of 6LoWPAN Implementations on Wireless Sensor Networks. In *Sensor Technologies and Applications, 2009. SENSORCOMM '09. Third International Conference on*, page 689–692.

Mischke, J. and Stiller, B. (2004). Peer-to-Peer and Distributed Search in Fixed and Mobile Environments. In *HICSS*.

Montenegro, G., Kushalnagar, N., Hui, J., and Culler, D. (2007). Transmission of IPv6 Packets over IEEE 802.15.4 Networks. RFC 4944 (Proposed Standard). Updated by RFCs 6282, 6775.

Moore, J. (2010). Packedobjects Reference Manual. http://zedstar.org/packedobjects/.

Moore, J., Bagale, J., and Kheirkhahzadeh, A. (2014a). XML corpus. https://github.com/jnbagale/phd-xml-corpus.

Moore, J., Bagale, J., Kheirkhahzadeh, A., and Komisarczuk, P. (2012). Fingerprinting Seismic Activity across an Internet of Things. In *New Technologies, Mobility and Security (NTMS), 2012 5th International Conference on*, pages 1–6.

Moore, J., Collins, T., and Shrestha, S. (2010). An Open Architecture for Detecting Earthquakes Using Mobile Devices. *Communications and Mobile Computing, International Conference on*, 1:437–441.

Moore, J., Kheirkhahzadeh, A., and Bagale, J. (2013). Domain-Specific XML Compression. In *Data Compression Conference (DCC), 2013*, pages 510–510.

Moore, J., Kheirkhahzadeh, A., and Bagale, J. (2014b). Towards Markup-Aware Text Compression. In *Data Compression Conference (DCC), 2014*, pages 417–417.

Mulligan, G. (2007). The 6LoWPAN Architecture. In *Proceedings of the 4th Workshop on Embedded Networked Sensors*, EmNets '07, page 78–82, New York, NY, USA. ACM.

Mulligan, G. and Bormann, C. (2005). IPv6 over Low power WPAN. http://www.ietf.org/wg/concluded/6lowpan.html.

Nakamura, Y. (2004). UrEDAS, urgent earthquake detection and alarm system, now and future. In *Proceedings of the 13th world conference on earthquake engineering*.

Odroid (2014). Odroid smart power measurement. http://odroid.com/dokuwiki/doku.php?id=en:odroidsmartpower.

OGC (2007). Sensor Markup Language (SensorML). http://www.opengeospatial.org/standards/sensorml.

Olivieri, M., Allen, R., and Wurman, G. (2008). The potential for earthquake early warning in Italy using ElarmS. *Bulletin of the Seismological Society of America*, 98(1):495.

Olsson, J. (2014). 6LoWPAN demystified. http://www.ti.com.cn/cn/lit/wp/swry013/swry013.pdf.

Orestis, A., Dimitrios, A., and Ioannis, C. (2012). Towards integrating IoT devices with the Web. In *Emerging Technologies Factory Automation (ETFA), 2012 IEEE 17th Conference on*, page 1–4.

Petersen, S. and Carlsen, S. (2011). WirelessHART Versus ISA100.11a: The Format War Hits the Factory Floor. *Industrial Electronics Magazine, IEEE*, 5(4):23–34.

Pi, R. (2012). Raspberry Pi model B. http://www.raspberrypi.org/products/model-b/.

Podnar Zarko, I., Antonic, A., and Pripužic, K. (2013). Publish/subscribe middleware for energy-efficient mobile crowdsensing. In *Proceedings of the 2013 ACM conference on Pervasive and ubiquitous computing adjunct publication*, UbiComp '13 Adjunct, page 1099–1110, New York, NY, USA. ACM.

Postel, J. (1980). User Datagram Protocol. RFC 768 (INTERNET STANDARD).

RabbitMQ (2011). RabbitMQ. http://www.rabbitmq.com/.

Richards, M., Monson-Haefel, R., and Chappell, D. A. (2009). *Java message service*. " O'Reilly Media, Inc.".

Rooney, S. and Garces-Erice, L. (2007). Messo & Preso Practical Sensor-Network Messaging Protocols. In *Universal Multiservice Networks, 2007. ECUMN'07. Fourth European Conference on*, page 364–376. IEEE.

Seddik-Ghaleb, A., Ghamri-Doudane, Y., and Senouci, S.-M. (2006). A performance study of TCP variants in terms of energy consumption and average goodput within a static ad hoc environment. In *Proceedings of the 2006 international conference on Wireless communications and mobile computing*, page 503–508. ACM.

Shen, C.-Y., Yang, D.-N., and Chen, M.-S. (2012). Collaborative and Distributed Search System with Mobile Devices. *IEEE Trans. Mob. Comput.*, 11(10):1478–1493.

Shevat, A. (2004). Distributed Enterprise Messaging with MantaRay. http://onjava.com/pub/a/onjava/2004/12/08/mantaray.html.

Singh, D., Tiwary, U., Lee, H., and Chung, W. (2009). Global healthcare monitoring system using 6lowpan networks. In *Advanced Communication Technology, 2009. ICACT 2009. 11th International Conference on*, volume 1, page 113–117. IEEE.

Sinha, A., Wang, A., and Chandrakasan, A. P. (2000). Algorithmic Transforms for Efficient Energy Scalable Computation. In *Proceedings of the 2000 International Symposium on Low Power Electronics and Design*, ISLPED '00, page 31–36, New York, NY, USA. ACM.

Sivakumar, R. and Akyildiz, I. F. (2008). GARUDA: Achieving effective reliability for downstream communication in wireless sensor networks. *IEEE Transactions on Mobile Computing*, 7(2).

Snyder, B., Bosanac, D., and Davies, R. (2010). Activemq in action. *MEAP, early access edition edition*.

Song, J., Han, S., Mok, A. K., Chen, D., Lucas, M., and Nixon, M. (2008). WirelessHART: Applying wireless technology in real-time industrial process control. In *Real-Time and Embedded Technology and Applications Symposium, 2008. RTAS'08. IEEE*, page 377–386. IEEE.

Srinivasan, K., Dutta, P., Tavakoli, A., and Levis, P. (2006). Some implications of low power wireless to ip networking. In *Proc. of the ACM Hot-Nets Conf*, page 31–37. Citeseer.

Stanton, J. (2002). Users Guide to Spread. www.spread.org/docs/guide/users_guide.pdf.

Swaroop, V. and Shanker, U. (2010). Mobile distributed real time database systems: A research challenges. In *Computer and Communication Technology (ICCCT), 2010 International Conference on*, page 421–424.

Szalapski, T., Madria, S., and Linderman, M. (2012). TinyPack XML: Real time XML compression for wireless sensor networks. In *Wireless Communications and Networking Conference (WCNC), 2012 IEEE*, page 3165–3170.

Tam, M.-C., Smith, J. M., and Farber, D. J. (1990). A taxonomy-based comparison of several distributed shared memory systems. *ACM SIGOPS Operating Systems Review*, 24(3):40–67.

Tay, B. H. and Ananda, A. L. (1990). A survey of remote procedure calls. *ACM SIGOPS Operating Systems Review*, 24(3):68–79.

Taysi, Z. C., Guvensan, M. A., and Melodia, T. (2010). TinyEARS: Spying on House Appliances with Audio Sensor Nodes. In *Proceedings of the 2Nd ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Building*, BuildSys '10, pages 31–36, New York, NY, USA. ACM.

Tsiftes, N. and Dunkels, A. (2011). A Database in Every Sensor. In *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*, SenSys '11, page 316–332, New York, NY, USA. ACM.

Veillard, D. (1999). The XML C parser and toolkit of Gnome- Libxml. http://xmlsoft.org.

W3C (1996). Extensible Markup Language (XML). http://www.w3.org/XML.

Wang, L. and Manner, J. (2009). Evaluation of data compression for energy-aware communication in mobile networks. In *Cyber-Enabled Distributed Computing and Knowledge Discovery, 2009. CyberC '09. International Conference on*, page 69–76.

Werner-Allen, G., Lorincz, K., Johnson, J., Lees, J., and Welsh, M. (2006). Fidelity and yield in a volcano monitoring sensor network. In *Proceedings of the 7th symposium on Operating systems design and implementation*, page 381–396. USENIX Association.

Xu, R., Li, Z., Wang, C., and Ni, P. (2003). Impact of data compression on energy consumption of wireless-networked handheld devices. In *Distributed Computing Systems, 2003. Proceedings. 23rd International Conference on*, page 302–311.

Ye, W., Heidemann, J., and Estrin, D. (2002). An energy-efficient MAC protocol for wireless sensor networks. In *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 3, page 1567–1576. IEEE.

Yibo, C., Hou, K.-m., Zhou, H., Shi, H.-L., Liu, X., Diao, X., Ding, H., Li, J.-J., and de Vaulx, C. (2011). 6LoWPAN Stacks: A Survey. In *Wireless Communications, Networking and Mobile Computing (WiCOM), 2011 7th International Conference on*, page 1–4.

ZigBee (2006a). ZigBee Alliance. http://zigbee. org/zigbeealliance/.

ZigBee (2006b). ZigBee Specification. http://www.zigbee.org.

ZigBee (2013). ZigBee IP. http://zigbee. org/zigbee-for-developers/network-specifications/zigbeeip/.

Appendix A

Messaging Protocols

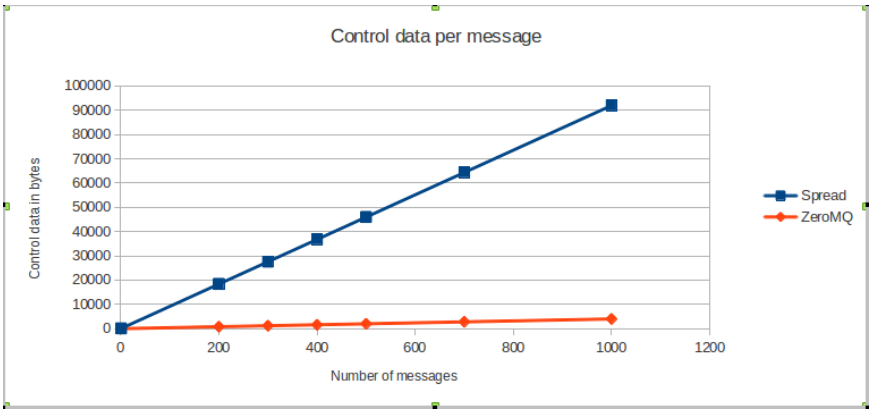

Fig. A.1 Graphical representation of control data overhead as a function of number of messages

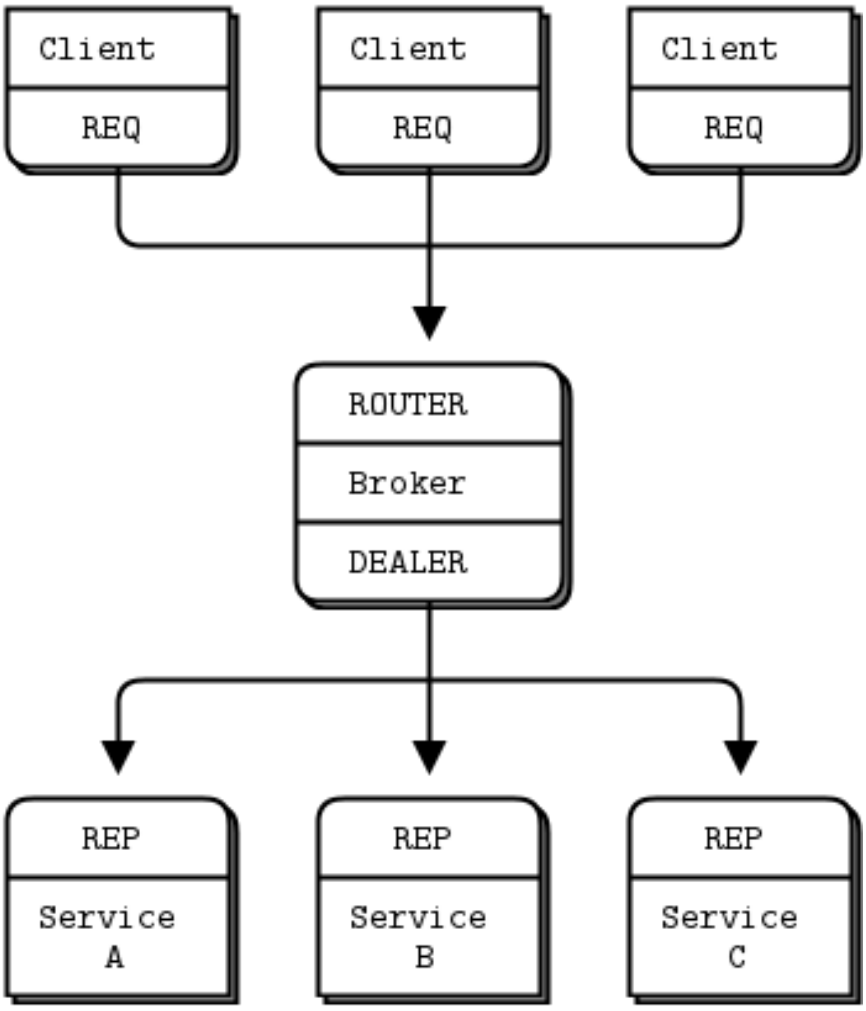

Fig. A.2 Request reply model with broker (Hintjens, 2011)

Appendix B

PackedobjectsD

B.1 XML and Schema for product search system

Listing B.1 XML Schema for product search system

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:include schemaLocation="packedobjectsDataTypes.xsd" />
  <!-- User defined types  -->

  <xs:simpleType name="categoryType">
    <xs:restriction base="enumerated">
      <xs:enumeration value="Motors"/>
      <xs:enumeration value="Fashion"/>
      <xs:enumeration value="Electronics"/>
      <xs:enumeration value="Home and Garden"/>
      <xs:enumeration value="Sporting Goods"/>
      <xs:enumeration value="Toys and Hobbies"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="conditionType">
    <xs:restriction base="enumerated">
      <xs:enumeration value="new"/>
      <xs:enumeration value="used"/>
      <xs:enumeration value="any"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:complexType name="databaseType">
    <xs:sequence>
      <xs:element name="product" maxOccurs="unbounded">
        <xs:complexType>
          <xs:sequence>
```

```
            <xs:element name="title">
              <xs:simpleType>
                <xs:restriction base="string">
                  <xs:maxLength value="100"/>
                </xs:restriction>
              </xs:simpleType>
            </xs:element>
            <xs:element name="category" type="categoryType"/>
            <xs:element name="condition" type="conditionType"/>
            <xs:element name="description">
              <xs:simpleType>
                <xs:restriction base="string">
                  <xs:maxLength value="100" />
                </xs:restriction>
              </xs:simpleType>
            </xs:element>
            <xs:element name="price" type="decimal"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="searchType">
    <xs:sequence>
      <xs:element name="product-title" type="string"/>
      <xs:element name="max-price" type="decimal"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="responseType">
    <xs:sequence>
      <xs:element name="responder-id" type="string"/>
      <xs:element name="product-title" type="string"/>
      <xs:element name="price" type="decimal"/>
      <xs:element name="category" type="categoryType"/>
      <xs:element name="condition" type="conditionType"/>
          <xs:element name="description" type="string"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="messageType">
    <xs:choice>
      <xs:element name="database" type="databaseType"/>
      <xs:element name="response" type="responseType"/>
      <xs:element name="search" type="searchType"/>
    </xs:choice>
  </xs:complexType>
```

```
  <xs:element name="products">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="message" type="messageType"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Listing B.2 XML database for product search system

```
<products>
  <message>
    <database>
      <product>
        <title>ipad</title>
        <category>Electronics</category>
        <condition>new</condition>
        <description>new iPad 2 16GB Wi-Fi Black</description>
        <price>219.99</price>
      </product>
      <product>
        <title>iphone</title>
        <category>Electronics</category>
        <condition>used</condition>
        <description>new iPhone 5 8GB White</description>
        <price>324.99</price>
      </product>
      <product>
        <title>galaxy s3</title>
        <category>Electronics</category>
        <condition>used</condition>
        <description>use samsung galaxy with usb</description>
        <price>124</price>
      </product>
      <product>
        <title>sony tv</title>
        <category>Electronics</category>
        <condition>used</condition>
        <description>50 inches 3d with 4 glasses</description>
        <price>567</price>
      </product>
      <product>
        <title>radio</title>
        <category>Electronics</category>
        <condition>used</condition>
        <description>dab digital</description>
        <price>78</price>
```

```
</product>
<product>
  <title>bose</title>
  <category>Electronics</category>
  <condition>used</condition>
  <description>3d cinematic with 5 speakers</description>
  <price>200</price>
</product>
<product>
  <title>apple tv</title>
  <category>Electronics</category>
  <condition>used</condition>
  <description>with packaged box</description>
  <price>50</price>
</product>
<product>
  <title>blackberry</title>
  <category>Electronics</category>
  <condition>used</condition>
  <description>black blackberry bold</description>
  <price>130</price>
</product>
<product>
  <title>macbook</title>
  <category>Electronics</category>
  <condition>used</condition>
  <description>13 inches macbook air</description>
  <price>700</price>
</product>
<product>
  <title>heater</title>
  <category>Electronics</category>
  <condition>used</condition>
  <description>skirting heater</description>
  <price>24.99</price>
</product>
<product>
  <title>cable</title>
  <category>Electronics</category>
  <condition>used</condition>
  <description>2 feet hdmi cable</description>
  <price>5</price>
</product>
<product>
  <title>charger</title>
  <category>Electronics</category>
  <condition>used</condition>
  <description>iPhone 5 charger</description>
```

```
        <price>5</price>
      </product>
      <product>
        <title>tennis racket</title>
        <category>Sporting Goods</category>
        <condition>new</condition>
        <description>still in package</description>
        <price>56</price>
      </product>
      <product>
        <title>tennis ball</title>
        <category>Sporting Goods</category>
        <condition>new</condition>
        <description>12 balls in the pack</description>
        <price>10</price>
      </product>
    </database>
  </message>
</products>
```

Listing B.3 Query XML for product search system

```
<?xml version="1.0" encoding="UTF-8"?>
<products>
  <message>
    <search>
      <product-title>ipad</product-title>
      <max-price>417.83</max-price>
    </search>
  </message>
</products>
```

Listing B.4 Response XML for product search system

```
<?xml version="1.0" encoding="UTF-8"?>
<products>
  <message>
    <response>
      <responder-id>417737407</responder-id>
      <product-title>ipad</product-title>
      <price>219.99</price>
      <category>Electronics</category>
      <condition>new</condition>
      <description>new iPad 2 16GB Wi-Fi Black</description>
    </response>
  </message>
</products>
```

B.2 Encode and Decode data

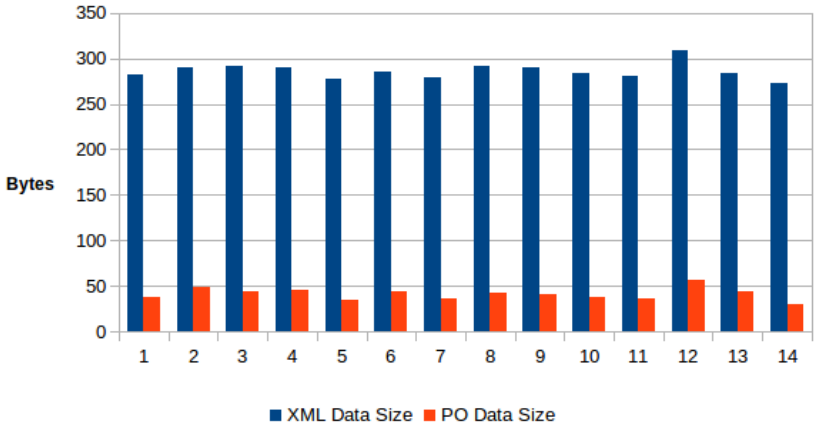

Fig. B.1 Responder Encode data size comparison

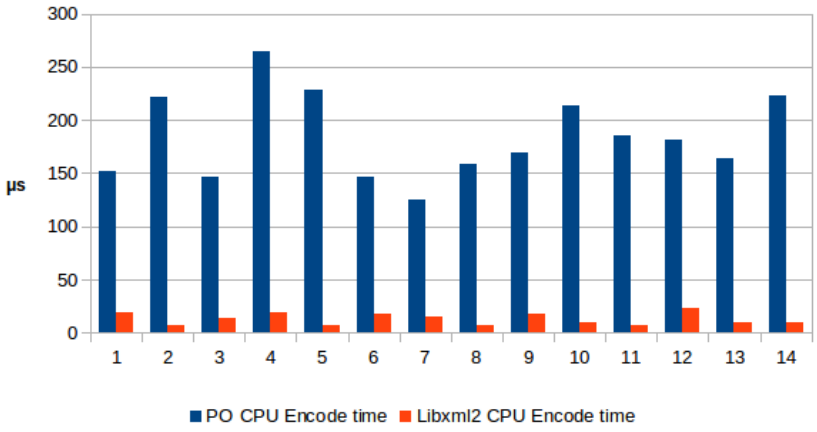

Fig. B.2 Responder Encode CPU time comparison

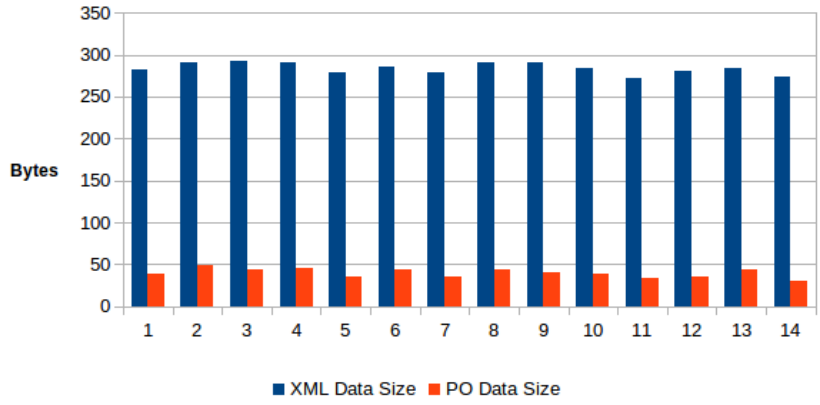

Fig. B.3 Searcher Decode data size comparison

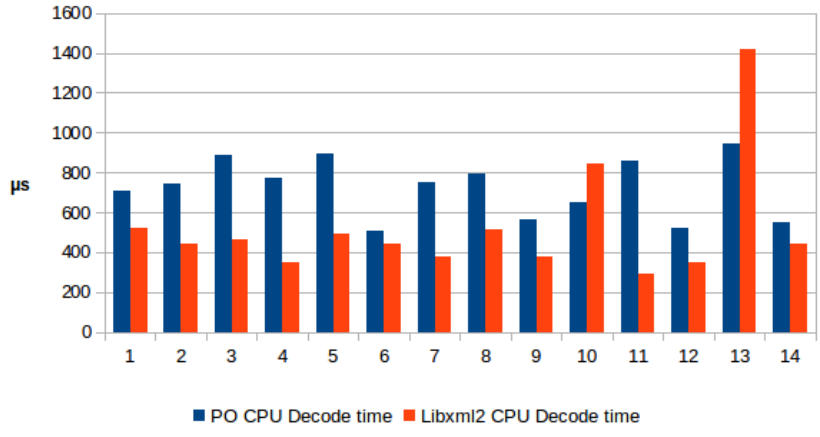

Fig. B.4 Searcher Decode CPU time comparison